# Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System

**Badrul M. Sarwar[*], Joseph A. Konstan[*†], Al Borchers[*], Jon Herlocker[*], Brad Miller[†], and John Riedl[*†]**

[*]GroupLens Research Project
Dept. of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455

[†]Net Perceptions, Inc
11200 West 78th Street
Suite 300
Minneapolis, MN 55344-3814

{sarwar,konstan,borchers,herlocker,riedl}@cs.umn.edu; bmiller@netperceptions.com

**ABSTRACT**

Collaborative filtering systems help address information overload by using the opinions of users in a community to make personal recommendations for documents to each user. Many collaborative filtering systems have few user opinions relative to the large number of documents available. This sparsity problem can reduce the utility of the filtering system by reducing the number of documents for which the system can make recommendations and adversely affecting the quality of recommendations.

This paper defines and implements a model for integrating content-based ratings into a collaborative filtering system. The filterbot model allows collaborative filtering systems to address sparsity by tapping the strength of content filtering techniques. We identify and evaluate metrics for assessing the effectiveness of filterbots specifically, and filtering system enhancements in general. Finally, we experimentally validate the filterbot approach by showing that even simple filterbots such as spell checking can increase the utility for users of sparsely populated collaborative filtering systems.

**Keywords**

Collaborative filtering, information filtering, content analysis, recommendation systems, social filtering, GroupLens Research, information filtering agents.

**INTRODUCTION**

Each day, more books, research papers, television programs, Internet discussion postings, and web pages are published than any individual human can hope to review, let alone read and understand. To cope with information overload, we try different approaches to separate the interesting and valuable information from the rest.



CSCW 98 Seattle Washington USA



Historically, this process was placed in the hands of editors and publishers—people given the responsibility for reviewing many documents and selecting the ones worthy of publication. Even today, we rely heavily on newspaper editors, moderators of discussion lists, journal editors and review boards. We also often read the opinions of movie, restaurant, and television critics to decide how to spend our finite time and money.

Professional human reviewers do not solve all problems, however. Often, individuals' information needs and tastes differ enough to make a small number of editors ineffective. Also, the number of documents in the web, in research libraries, and in archives of discussions has grown so large as to defy systematic editing by individual editors. Accordingly, researchers have developed a wide range of systems that bring the power of computation to the problem of selecting, for each individual, the information he or she considers valuable from the enormous amount of available information.

*Information retrieval* (IR) systems allow users to express queries to select documents that match a topic of interest. IR systems may index a database of documents using the full text of the document or only document abstracts. Sophisticated systems rank query results using a variety of heuristics including the relative frequency with which the query terms occur in each document, the adjacency of query terms, and the position of query terms. IR systems also may employ techniques such as term stemming to match words such as "retrieve," "retrieval," and "retrieving." [15] IR systems are generally optimized for ephemeral interest queries, such as looking up a topic in the library. [3] In the Internet domain, popular IR systems include AltaVista (www.altavista.digital.com) for web pages and DejaNews (www.dejanews.com) for discussion list postings.

*Information filtering* (IF) systems use many of the same techniques as IR systems, but are optimized for long-term information needs from a stream of incoming documents. Accordingly, IF systems build user profiles to describe the documents that should (or should not) be presented to users. Simple examples of IF systems include "kill files" that are

used to filter out advertising or flames (i.e., attack messages) and e-mail filtering software that sorts e-mail into priority categories based on the sender, the subject, and whether the message is personal or sent to a list. More complex IF systems provide periodic personalized digests of material from sources such as news wires, discussion lists, and web pages [3]

One embodiment of IF techniques is software *agents*. These programs exhibit a degree of autonomous behavior, and attempt to act intelligently on behalf of the user for whom they are working. Agents maintain user interest profiles by updating them based on feedback on whether the user likes the items selected by the current profile. Research has been conducted in various feedback generation techniques, including probabilistic models, genetic algorithms and neural network based learning algorithms [2]. *NewT* is a filtering agent for Usenet news based on genetic algorithm learning techniques [8]. It performs full text analysis of articles using vector-space technique. *Amalthaea* is a multi-agent system for personalized filtering, discovery and monitoring of information sources in the *World Wide Web* domain [11].

IR and IF systems can be extremely effective at identifying documents that match a topic of interest, and at finding documents that match particular patterns (e.g., discarding e-mail with the phrase "Make Money Fast" in the title). Unlike human editors, however, these systems cannot distinguish between high-quality and low-quality documents on the same topic. As the number of documents on each topic continues to grow, even the set of relevant documents will become too large to review (e.g., who has time to read every technical report with CSCW in the keyword list?). For some domains, therefore, the most effective filters must incorporate human judgements of quality.

*Collaborative filtering* (CF) systems recommend documents to a user based on the opinions of other users. In their purest form, CF systems do not consider the content of the documents at all, relying exclusively on the judgement of humans as to whether the document is valuable. In this way, collaborative filtering attempts to recapture the cross-topic recommendations that are common in communities of people.

*Tapestry* [4], one of the first computer-based collaborative filtering systems, was designed to support a small, close-knit community of users. Users could filter all incoming information streams, including e-mail and Usenet news articles. When users evaluated a document, they could annotate it with text, with numeric ratings, and with boolean ratings. Other users could form queries such as "show me the documents that Mary annotated with 'excellent' and Jack annotated with 'Sam should read.'" A similar approach is used in Maltz and Ehrlich's *active collaborative filtering* [9], which provides an easy way for users to direct recommendations to their friends and colleagues through a *Lotus Notes* database.

Collaborative filtering for large communities cannot depend on each person knowing the others. Several systems use statistical techniques to provide personal recommendations of documents by finding a group of other users, known as *neighbors*, that have a history of agreeing with the target user. Once a neighborhood of users is found, particular documents can be evaluated by forming a weighted composite of the neighbors' opinions of that document. Similarly, a user can request recommendations for a set of documents to read and the system can return a set of documents that is popular within the neighborhood. These statistical approaches, known as *automated collaborative filtering*, typically rely upon *ratings* as numerical expressions of user preference.

Several ratings-based automated collaborative filtering systems have been developed. The GroupLens Research[1] system [6,13] provides an pseudonymous collaborative filtering solution for Usenet news and movies. *Ringo* [16] *and Video Recommender* [5] are email and web systems that generate recommendations on music and movies respectively, suggesting collaborative filtering to be applicable to many different types of media. Indeed, commercial applications of ratings-based collaborative filtering now exist in a variety of domains including books, music, grocery products, dry goods, and information.

While collaborative filtering has been a substantial success, there are several problems that researchers and commercial applications have identified:

**The early-rater problem.** A collaborative filtering system provides little or no value when a user is the first one in his neighborhood to enter a rating for an item. Current collaborative filtering systems depend on the altruism of a set of users who are willing to rate many items without receiving many recommendations. Economists have speculated that even if rating required no effort at all, many users would choose to delay considering items to wait for their neighbors to provide them with recommendations [1]. Without altruists, it might be necessary to institute payment mechanisms to encourage early ratings.

**The sparsity problem.** The goal of collaborative filtering systems is to help people focus on reading documents (or consuming items) of interest. In high-quantity, low-quality environments, such as Usenet news, users may cover only a tiny percentage of documents available (Usenet studies have shown a rating rate of about 1% in some areas; we can estimate that few people will have read and formed an opinion on even 1/10 of 1% of the over two million books available through the largest bookstores). On the one hand, this sparsity is the motivation behind filtering: most people

[1] GroupLens™ is a trademark of Net Perceptions, Inc., which holds exclusive rights to commercialize the results of the GroupLens Research project. Net Perceptions allows the University of Minnesota to use the name GroupLens Research to avoid name discontinuity in the project.

do not want to read most available information. On the other hand sparsity poses a computational challenge as it becomes harder to find neighbors and harder to recommend documents since few people have rated most of them.

Efforts have been made to overcome these problems in collaborative filtering system:

- **partitioning**. The GroupLens Research project showed that partitioning the ratings database by Usenet newsgroup resulted in somewhat higher accuracy and density, since not all users subscribed to all newsgroups. Even with partitioning, however, sparsity was still a problem.
- **dimensionality reduction**. Several researchers have been examining statistical techniques for compressing the dimensionality of the database. These techniques, which include general clustering, singular value decomposition, factor analysis, and others appear promising, but none has yet been demonstrated to solve the sparsity problem.
- **implicit ratings**. Several systems attempt to increase the number of ratings entered by observing user behavior. The GroupLens Research system determined that time spent reading a Usenet news article was an effective rating measure [10]. PHOAKS found that URLs mentioned in Usenet postings could be filtered to detect recommendations [17]. Other systems have examined user history or watch user behavior[14,12]. At the extreme, the MovieLens system was able to reduce start-up sparsity somewhat by incorporating several million pre-existing ratings.

We should point out that content-based approaches used in IF and agent systems are less directly affected by these problems because they use content analysis techniques that apply across all documents. For example, a filter that gives high scores to articles with the word "baseball" in them, can give a score to a new article before anyone has rated it. To exploit the advantages of content analysis, Fab implements a hybrid content-based collaborative system for recommending Web pages [2]. In Fab user profiles are maintained by using content analysis. The profiles are directly compared to determine similarity between users to support collaborative recommendation.

In this paper, we investigate another hybrid approach to addressing the rating sparsity and early rater problems. This approach incorporates semi-intelligent filtering agents called filterbots into a ratings-based collaborative filtering system.

## RESEARCH APPROACH: THE FILTERBOT CONCEPT

Our approach to addressing the ratings sparsity and early rater problems is to incorporate non-collaborative information filtering techniques into a collaborative filtering system. We introduce these techniques through the creation of *filterbots*—automated rating robots that evaluate new documents as soon as they are published and enter ratings for those documents. We chose this model because we found that it is appealingly simple from both the collaborating filtering system's and the filterbot author's point of view.

The **collaborative filtering system** treats a filterbot as another ordinary user, albeit a prolific and generous one that enters many ratings but doesn't request predictions. The collaborative filtering engine need not even know whether users are filterbots or humans.

The **filterbot author** writes a filterbot just like an information filtering agent. This agent is called whenever new documents arrive, and it returns a numeric rating (1 through 5 in our system). The filterbot author need not be concerned with the use of the filterbot in a collaborative filtering system.

There are other approaches to merging content filtering with collaborative filtering, including the "communicating agents" model proposed by Maes [8] and the correlating profiles model in Fab [2]. We found the filterbot model more appealing than alternative models of integrating information filtering techniques with collaborative filtering ones because the collaborative filtering engine already includes a filter to personalize the weight assigned to each filterbot. If a user agrees consistently with a filterbot, that filterbot is accorded a high weight for the user. If a user's ratings do not correlate well with a filterbot's, that filterbot is not used in generating recommendations and predictions for that specific user.

An implication of this design is that we can employ a wide range of algorithms in filterbots without concern that an algorithm would have a detrimental effect on individual users. By comparison, a system that employs a filter across all users (e.g., system-level advertisement detection and kill files) indiscriminately filters out content for both those who are annoyed by advertising and those interested in learning about new products.

In this work, we report on a set of filterbots using very simple algorithms such as spelling correctness and article length. By demonstrating the value of the filterbot concept on simple algorithms, we hope to encourage people who have insight into factors related to user preferences in collaborative filtering systems to build their own filterbots. Authors don't need know anything about collaborative filtering; they just need an idea for a strategy to automatically rate items. Write it; throw it in; and watch people benefit!

We also recognize, but have not yet implemented, the potential for incorporating learning agents as filterbots in a collaborative filtering system. The collaborative filtering system might receive ratings from one or several filterbots per user, and the users would benefit from having access to the learned preferences of each agent, again relying upon only those filterbots with whom they have agreed individually over time. Agents researchers can further this process by creating a community of agents under natural

selection rules, so agents that are ineffective are eliminated to create resources for variants of effective ones.

The rest of this paper presents our research design for assessing the value of filterbots in a collaborative filtering system, presents the results of the experiment, and discusses the limitations of our work and the implications for other researchers in collaborative filtering and information filtering.

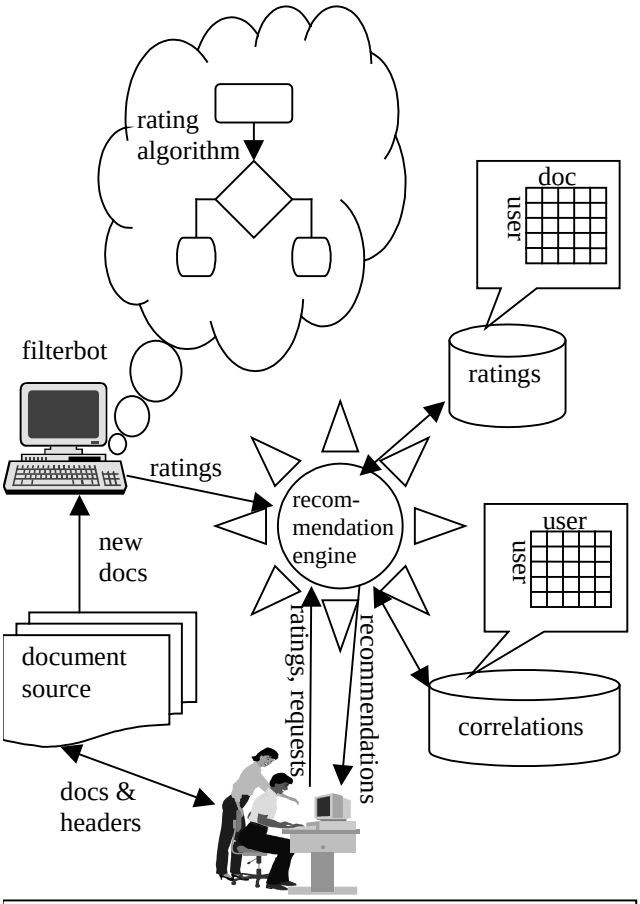


**Figure 1.** System architecture for a collaborative filtering system with live users and filterbots.

## RESEARCH DESIGN

### Architecture

The basic idea of collaborative filtering system is to help people collaborate to find items of interest from a large collection of items. In this section, we describe the architectural framework of a collaborative filtering system that can support the incorporation of filterbots. This architecture is based on using the GroupLens Research Recommendation Engine, but a similar architecture would work with any collaborative filtering engine that works using ratings. An overview of this architecture is shown in figure 1.

In general, collaborative filtering systems employ a client-server architecture. CF clients are programs that present the user with an interface for browsing documents. Example clients include news readers and web browsers. These client applications communicate with document servers to retrieve documents for the user (e.g., news servers, web servers, databases). The clients use a well-known document server API to request these items (e.g., NNTP, HTTP). Clients also communicate with a *recommendation engine* server through its API. Calls are provided to enter ratings for particular documents, to request recommendations for documents to request, or to evaluate a set of useful documents.

In the original GroupLens Research trial, the clients were Usenet news readers that had been specially modified to connect to the GroupLens Research server. The readers, which included gnus, xrn, and tin, were adapted to request predicted values from the GroupLens Research server before displaying article subjects to each user. As the user read a newsgroup, she could enter ratings. Those ratings were stored by the reader and sent to the server at the end of the newsgroup.

The GroupLens Research engine stored two sets of data: user ratings of news articles and user-user correlation information. From the correlation table, the recommendation engine can quickly identify the neighborhood of similar users for prediction purposes. A prediction is calculated by returning a weighted average of normalized ratings, as reported in [13].

Filterbots are incorporated into this framework as follows:

- They request (or subscribe to) new items from the document source.
- They apply the rating algorithm to each document that arrives.
- When the algorithm returns a rating, they submit that rating to the recommendation engine.

As we implemented them, the filterbots poll the Usenet news server to request new items. News clients already have mechanisms for detecting which articles are new (a file that stores the range of read items for each newsgroup). The ratings algorithms, which are described below, perform simple content analysis and produce ratings for all articles.

### Hypothesis

Because of the ratings sparsity and early rater problems, collaborative filtering systems are often only able to offer users predictions and recommendations for a small subset of the documents available. The filterbot framework provides an augmentation that should improve the value of collaborative filtering systems. By integrating content filtering into collaborative filtering, filterbots should increase the utility of the system for users who agree with the filterbots while not affecting other users. Accordingly, we propose the following hypothesis:

**H1:** *Adding content-based filterbots into a collaborative filtering system improves utility for human users.*

We should clearly state that we are not evaluating the value of filterbots *without* human ratings, for the simple reason

that human ratings are necessary for computing the agreement among the users and between users and filterbots.

### Experiment Design

We implemented three different filterbots: SpellCheckerBot, IncludedMsgBot and LengthBot. We conducted our experiments by incorporating these filterbots individually into the GroupLens Research collaborative filtering system. The filterbots fetch and analyze articles from the *Usenet news* server and send the ratings directly to the *GroupLens* recommendation engine using the *GroupLens client library API.* Our filterbots were applied to five different newsgroups selected to be representative of many of the different types of newsgroups on Usenet. The newsgroups are: (1) *rec.humor,* a low-quality high-volume entertainment newsgroup, (2) *rec.food.recipes,* a moderated newsgroup with no discussion, (3) *mn.general,* a local discussion newsgroup, (4) *comp.lang.perl.misc,* a technical discussion group (5) *comp.os.linux.announce,* a technical announcement group.

Next we describe the design of each filterbot.

**SpellCheckerBot** rates articles based on the proportion of spelling errors present in the article text. It uses the *spell* utility of the *unix* system as its spell-checking engine. *Spell* uses its own dictionary to check the words in a document and dumps the words not found the dictionary as misspelled words. As a result, any correctly spelled word that is not present in *spell*'s dictionary will be taken as a misspelled word. Such words include widely used colloquial expressions, word abbreviations, acronyms, technical terms, proper nouns and so on. Using *spell*'s internal dictionary will incorrectly count these words as spelling errors. The addition of an *auxiliary dictionary* solves this problem. This auxiliary dictionary contains a list of known correct words and is used by *spell* in addition to its own dictionary. Since Usenet newsgroups carry discussions on different topics, an auxiliary dictionary intended for a particular newsgroup will not, in general, be applicable to another newsgroup. For example, the word *gzipped* is added to the auxiliary dictionary for *comp.lang.perl.misc* newsgroup but is not a suitable entry into the auxiliary dictionary for the *rec.food.recipes* newsgroup, where the word *canola* would be suitable.

We created the auxiliary dictionary for each newsgroup by running the *spell* program on the message bodies of each article, collecting the set of words that spell did not recognize. We then hand-reviewed all of the terms that were frequently misspelled to determine whether they should be added to the dictionary, or were instead simply common misspellings (the word "receive" was commonly misspelled, for example). For real-world use, this start-up phase would be performed once, with an incremental process that could add new words to the dictionary as they come into use in a newsgroup.

Once the dictionary was created, the filterbot processed each message by:

1. stripping off headers, signatures, and included text from prior messages ,
2. running the *spell* program to count the number of misspelled words,
3. counting the number of words in the message body,
4. converting the percentage of misspelled words into a rating on a scale of 1 through 5, and
5. submitting the rating to the recommendation engine.

To avoid confounding variables in the experiment, we chose to establish a mapping between misspelling percentage and rating that would result in a ratings distribution that closely approximated the human ratings distribution for that newsgroup. Prior experience suggests that correlation-based collaborative filtering algorithms are not very sensitive to individual differences in rating distribution, but keeping the distribution the same allowed us to avoid depending on those experiences. The same mapping strategy is used in the other filterbots.

**IncludedMsgBot** rates each article based on the percentage of text quoted from other articles. Replies to discussion threads often include some or all of the message being replied to. In some cases, as a thread continues the amount of included text grows substantially. Our experience and discussion with users suggested that many users dislike long messages with little new content.

IncludedMsgBot searches for this type of message, giving low ratings to articles with large amounts of included text and high ratings to articles with little included text. The filterbot:

1. separates out lines with a prefix of ">" -- most news posting software uses this convention to mark included text, and a hand inspection of the text confirmed that it was a useful heuristic for these newsgroups;
2. counts lines of new text and lines of included text;
3. computes the ratio of included text lines to total lines, and coverts that ratio to a rating on the scale of 1 through 5; and
4. submits the rating to the recommendation engine.

**LengthBot** rates articles based on the hypothesis that Usenet readers value brevity. After stripping off headers, signatures, and included text, LengthBot counts the number of words in the article body and converts the length into a rating on the scale of 1 to 5. Shorter articles receive higher ratings and longer ones receive lower ratings.

### Analysis of Metrics

In our hypothesis, we use the concept of "improved utility." Given the goal of collaborative filtering systems—helping users more effectively identify the content they want—we define utility to include two dimensions: *coverage* and *accuracy*.

*Coverage* is a measure of the percentage of items for which a recommendation system can provide recommendations. A low coverage value indicates that the user must either forego a large number of items, or evaluate them based on criteria other than recommendations. A high coverage value indicates that the recommendation system provides assistance in selecting among most of the items.

A basic coverage metric is the percentage of items for which predictions are available. This metric is not well-defined, however, since it may vary per user, depending on the user's ratings and neighborhoods. To address this problem, we use a usage-centric coverage measure that asks the question: "Of the items evaluated by the user, what percentage of the time did the recommendation system contribute to the evaluation process?" More formally, for every rating entered by each user, was the system able to make a recommendation for that item immediately prior to it being rated? We compute the percentage of recommendation-informed ratings over total ratings as our coverage metric.

*Accuracy* has been measured in many different ways in prior research. The two general approaches used are *statistical recommendation accuracy* and *decision-support accuracy*. [16]

*Statistical recommendation accuracy* measures the closeness between the numerical recommendations provided by the system and the numerical ratings entered by the user for the same items. Three common metrics used are *Correlation*, *Mean Absolute Error* (MAE), and *Root Mean Squared Error* (RMSE).

**Correlation** is a statistical measure of agreement between two vectors of data. We use the standard Pearson correlation coefficient as a measure of linear agreement between the two vectors. A higher correlation value indicates more accurate recommendations.

**MAE** is a measure of the deviation of recommendations from their true user-specified values. The lower the *MAE*, the more accurately the recommendation engine predicts user ratings.

**RMSE** is a measure of error that is biased to weigh large errors disproportionately more heavily than small errors. The intuition behind RMSE is that many recommendations that are off by .25 on a scale of 5 are better than a few ones off by 3 or 4. Like *MAE*, lower *RMSE* also indicates better accuracy.

*Decision-support accuracy* measures how effectively recommendations help a user select high-quality items. They are based on the observation that for many users, filtering is a binary process. The user either will or will not read the document or consume the article. In the Usenet news case, users make rapid decisions about whether to read an article, and the difference between a recommendation score of 4.0 and 4.5 is irrelevant if the user reads everything rated 4 and above. Similarly, the difference between 1 and 2 is irrelevant if either article will be skipped. Three measures of decision-support accuracy are *reversal rate*, *ROC sensitivity*, and *PRC sensitivity*.

**Reversal rate** is a measure of how often the system makes big mistakes that might undermine the confidence that a user has in the recommendation system. Low reversals refer to cases where the user strongly dislikes an item (i.e., gives a rating lower than a threshold $L$) and the system strongly recommends it with a high recommendation score (i.e., above a threshold $H$). High reversals are cases where the user strongly likes the item, but the system recommendation is poor (i.e., user rating > $H$, system recommendation < $L$).

**ROC sensitivity** is a measure of the diagnostic power of a filtering system. Operationally, it is the area under the receiver operating characteristic (ROC) curve—a curve that plots the sensitivity and specificity of the test [7]. Sensitivity refers to the probability of a randomly selected good item being accepted by the filter. Specificity is the probability of a randomly selected bad item being rejected by the filter. The ROC curve plots sensitivity (from 0 to 1) and *1 – specificity* (from 0 to 1), obtaining a set of points by varying the recommendation score threshold above which the article is accepted. The area under the curve increases as the filter is able to retain more good items while accepting fewer bad items. For use as a metric, we must determine which items are "good" and which are "bad." For that task, we use the user's own ratings. A rating of 4 or 5 is deemed to be a good item (signal), a rating of 1, 2, or 3 is deemed to be a bad item (noise). The ROC sensitivity measure therefore is an indication of how effectively the system can steer people towards highly-rated items and away from low-rated ones. Particularly important values are 1.0, the perfect filter, and 0.5, a random filter.

**PRC sensitivity** is a measure of the degree to which the system presents relevant information. In fact, it is the area under the precision-recall curve. Precision measures the percentage of selected documents that are relevant; recall measures the percentage of relevant documents that are selected. Hence, precision indicates how selective the system is, and recall indicates how thorough it is in finding valuable information. [15] We use as the domain of our metric the set of articles on which the user has recommendation-informed ratings. We plot a curve of different precision-recall pairs for different recommendation score thresholds, and take the area under that curve as a metric of PRC sensitivity. A higher value is more accurate, and a low value is less accurate.

For these experiments, we use ROC and PRC sensitivity as our primary accuracy metrics because they most closely match the goals of our experiments. We are more interested in whether adding filterbots to the GroupLens Research system helps users decide whether to read articles than in minimizing errors in areas that do not affect user decisions. Large reversals, the other decision-support metric, were too infrequent in this data set to use with confidence. As a sanity check, we did analyze our results using MAE and

RMSE; results were similar, which is in line with the finding of prior work that accuracy improvements tend to be reflected across the spectrum of metrics.

## EXPERIMENTS WITH FILTERBOTS

### Data

The data we used for these experiments is from the GroupLens Research trial of Winter 1996. During that seven-week trial, described in [10], we collected 47,569 ratings from over 250 users across many newsgroups. The newsgroups used for these experiments are a cross section of technical and recreational, moderated and unmoderated.

### Procedure

To test our hypothesis *H1* we need to get predictions from the *GroupLens* server both with and without filterbot ratings. For each newsgroup we created four files of data: one with user ratings only, and one each with the ratings of the three filterbots. Each record contained a user ID, newsgroup, message ID, and rating. To obtain base statistics for user-only recommendations, we followed this procedure:

1. Create an empty GroupLens database.
2. For each rating in the ratings file:
   a. request a recommendation for that user/newsgroup/message;
   b. record the returned recommendation or lack thereof; and
   c. submit the rating.
3. Compute coverage and accuracy statistics.

For the filterbot experiments, after step #1, we loaded all filterbot ratings into the database, and then proceeded with steps #2 and #3.

The experimental configuration uses Net Perceptions' commercial GroupLens Recommendation Engine version 2.2.2.5 configured to use a neighborhood size of 50 and no neighbor correlation threshold.

## RESULTS

**H1** hypothesizes that that adding filterbots into a collaborative filtering system will improve utility for users. To test this hypothesis, we look at experiments with several different Usenet newsgroups and several different filterbots. Since utility is a function of both item coverage and accuracy, we examine coverage, ROC sensitivity, and PRC sensitivity metrics in each newsgroup with each filterbot and without filterbots. If the coverage and accuracy both increase, then we can accept **H1**. If coverage increases and accuracy is unchanged, or accuracy increases with no change in coverage, we can also accept **H1**. However, if either coverage or accuracy decreases, we will be unable to accept **H1**. Because the effectiveness of filterbots may vary by newsgroup, we present the results separately for each newsgroup. Then we look at the value of each filterbot overall, and evaluate the hypothesis in general.

### Results by Newsgroup

#### mn.general

This newsgroup is a local unmoderated newsgroup with discussion and announcements on all topics from local events to finding reliable or inexpensive Internet service. We had 17 users who rated an average of 65 of the 559 articles in the newsgroup (for an average of 1.98 ratings per article). As table 1 shows, coverage improved somewhat for the newsgroup with each filterbot, but accuracy either decreased slightly or was inconclusive. Accordingly, we were unable to accept **H1** for **mn.general**.

**Table 1 : Results for mn.general newsgroup**

| Filterbots | Coverage (%) | ROC Sensitivity | PRC Sensitivity |
|---|---|---|---|
| No filterbot | 40.670 | 0.6937 | 0.2295 |
| SpellChecker | 43.155 | 0.6779 | 0.2075 |
| IncludedMsg | 46.056 | 0.7044 | 0.2180 |
| Length | 44.609 | 0.6719 | 0.2110 |

#### comp.lang.perl.misc

The newsgroup comp.lang.perl.misc is an unmoderated technical discussion group focused on the scripting language Perl. The *.misc* suffix indicates that this group receives primarily articles that do not fit into one of the other Perl newsgroups. We had 10 users who rated an average of 70 of the 627 articles in the newsgroup (for an average of 1.66 ratings per article). As table 2 shows, coverage and accuracy improved dramatically for the spell checking filterbot. Coverage increased by 85% for the other filterbots, with the included message algorithm having no significant effect on accuracy and the length algorithm having a very small positive effect. Given the success of the spell checking filterbot, we are able to accept **H1** for **comp.lang.perl.misc**.

**Table 2 : Results for comp.lang.perl.misc newsgroup**

| Filterbots | Coverage (%) | ROC Sensitivity | PRC Sensitivity |
|---|---|---|---|
| No filterbot | 7.010 | 0.6523 | 0.4698 |
| SpellChecker | 42.775 | 0.7448 | 0.6030 |
| IncludedMsg | 13.017 | 0.6400 | 0.4694 |
| Length | 13.180 | 0.6770 | 0.4981 |

#### comp.os.linux.announce

The newsgroup comp.os.linux.announce is a moderated technical discussion group that is used to make announcements to developers and users of the Linux operating system. We had 23 users who rated an average of 24 of the 421 articles in the newsgroup (for an average of 1.33 ratings per article). As table 3 shows, the length and spell checking filterbots both provided dramatic increases in coverage with moderate increases in accuracy. The included message filterbot increased coverage somewhat, but decreased accuracy. Given the success of the spell checking and length filterbots, we are able to accept **H1** for **comp.os.linux.announce**.

**Table 3 : Results for comp.os.linux.announce group**

| Filterbots | Coverage (%) | ROC Sensitivity | PRC Sensitivity |
|---|---|---|---|
| No filterbot | 14.874 | 0.6619 | 0.3234 |
| SpellChecker | 46.319 | 0.6822 | 0.3643 |
| IncludedMsg | 20.430 | 0.6117 | 0.3146 |
| Length | 48.745 | 0.7046 | 0.3686 |

### rec.food.recipes

The newsgroup rec.food.recipes is a moderated recreational group where contributors post recipes and occasional requests for recipes. We had 7 users who rated an average of 22 of the 92 articles in the newsgroup (for an average of 1.66 ratings per article). As table 4 shows, the spell checking filterbot greatly increased coverage and accuracy, with a particularly strong increase in PRC sensitivity. The included message filterbot provided a much smaller increase in coverage, but similarly impressive accuracy improvements. The length filterbot provided inconclusive accuracy results (worse ROC, better PRC) with an in-between increase in coverage. Based on the strength of the spell checking filterbot, we are able to accept **H1** for **rec.food.recipes**.

**Table 4 : Results for rec.food.recipes newsgroup**

| Filterbots | Coverage (%) | ROC Sensitivity | PRC Sensitivity |
|---|---|---|---|
| No filterbot | 22.222 | 0.6181 | 0.3902 |
| SpellChecker | 71.710 | 0.6601 | 0.6254 |
| IncludedMsg | 27.451 | 0.6667 | 0.5937 |
| Length | 42.763 | 0.5687 | 0.4570 |

### rec.humor

The newsgroup rec.humor is an unmoderated recreational group where contributors are expected to post jokes and other humorous material. It is a well-known high-noise newsgroup that is commonly cited as an example of a group where filtering is useful. We had 19 users who rated an average of 92 of the 1367 articles in the newsgroup (for an average of 1.27 ratings per article). As table 5 shows, all three filterbots provided dramatic increases in accuracy, and two of them also increased coverage by more than 200%. Accordingly, we are able to accept **H1** for **rec.humor**.

**Figure 2: Correlations between users and filterbots**

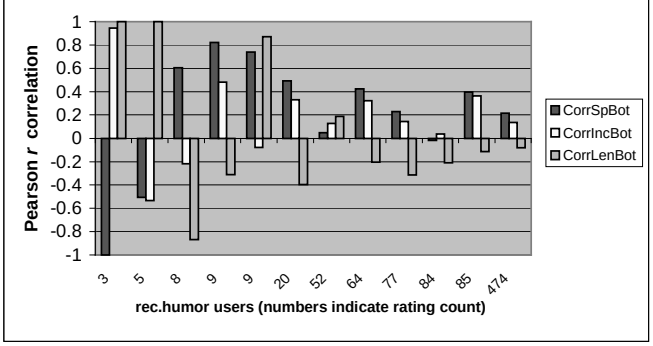


**Table 5: Results for rec.humor newsgroup**

| Filterbots | Coverage (%) | ROC Sensitivity | PRC Sensitivity |
|---|---|---|---|
| No filterbot | 15.384 | 0.4604 | 0.1253 |
| SpellChecker | 50.258 | 0.8081 | 0.3638 |
| IncludedMsg | 50.373 | 0.7228 | 0.3915 |
| Length | 16.657 | 0.7188 | 0.2487 |

Since rec.humor was the group with the largest combination of effect size and number of users, we decided to look more closely at the degree to which individual users agreed with the filterbots. Figure 2 shows rating correlations between the three filterbots and the twelve users who had rated enough articles to have correlations. The extreme correlations for five users reflect their small number of ratings, rather than any systematic agreement. Several users with large numbers of ratings have fairly high correlations, particularly with the spell checking filterbot. Of six users with more than 50 ratings: four have correlations greater than 0.2 with SpellCheckerBot (two at or above 0.4), two have correlations above 0.3 with IncludedMsgBot, and three have negative correlations stronger than –0.2 with LengthBot. Even the least-correlated user (the one with 52 ratings) had a correlation of 0.19 with one of the filterbots. We can use the user-filterbot correlation to determine which filterbots are most useful for a given set of users. Filterbots that are not valuable for a set of users can be turned off, to save computation. From the above diagram we see that our notion of "long messages are bad" may not be suitable for the users in the rec.humor newsgroup. A possible reason for the poor performance of this filterbot is that users actually prefer long messages in this newsgroup. We had disabled negative correlations for our study, since in previous work they led to worse results [16]. To rule out this possibility, we also tested a filterbot that rated using the rule "short messages are bad", but that filterbot performed even more poorly. Our conclusion is that length is not a good predictor of quality in rec.humor.

### Results by Filterbot

**SpellCheckerBot** shows very promising results. It provided both improved coverage and improved accuracy in four of the five newsgroups we tested. It appears that Usenet readers prefer articles with correct spelling. This does not necessarily imply that they care about spelling *per se*, but that something that they care about correlates well with spelling. For example, readers may value careful writing, simple vocabularies, etc.

**IncludedMsgBot** shows mixed results, providing improved coverage in all groups but accuracy improvements only in the recreational groups. One interpretation is that the nature of a group determines whether included content is good, bad, or neutral. Technical groups (and mn.general) often have discussions where readers appear to value the

context provided by included text. In these group, the best amount of included text is neither "more" nor "less" but simply "the right amount." Neither the recipe nor the humor group is supposed to have much included text at all. Rec.food.recipes is not to be used for discussion of recipes and rec.humor is not to be used for discussion of jokes. Accordingly, results in these groups may mostly reflect identifying and giving low ratings to out-of-place postings (for rec.food.recipes, the periodic "request for recipes" collection).

**LengthBot** showed benefits for rec.humor and comp.lang.perl.misc, and showed the best results by all measures in comp.os.linux.announce. Length was not useful in mn.general or rec.food.recipes. As with included text, this suggests that readers of different newsgroups value different attributes. Indeed, much of LengthBot's value in rec.humor came from negative correlations—people who apparently preferred longer articles.

### Overall Results

Based on all of the results presented above, we accept the hypothesis that content-analysis filterbots can improve user utility. In four of the five newsgroups, we found at least one filterbot that improved both coverage and accuracy, measured both by ROC and PRC. We analyzed statistical significance of the ROC values at 95% confidence level. SpellCheckerBot showed statistically significant improvements compared to the corresponding values obtained without using filterbots in three newsgroups, but in the mn.general and comp.os.linux.announce newsgroups the ROC values were not significantly different. IncludedMsgBot showed significantly higher ROC values in rec.humor and significantly lower ROC values in comp.os.linux.announce newsgroup. In the rest of the newsgroups ROC values did not change significantly. LengthBot provided significantly better results in two newsgroups and worse ROC values in rec.food.recipes. In mn.general and comp.lang.perl.misc newsgroups differences in ROC values were not statistically significant.

## DISCUSSION

These experiments demonstrate that simple content-analysis filterbots can help improve the coverage and accuracy of a collaborative filtering system. We recognize that there are several important limitations to this work, but also many exciting applications of it. In this section, we discuss both, along with some of our ideas for future work.

### Limitations

This study is one of the first on combining collaborative filtering techniques with syntactic filtering techniques. We therefore focussed on very simple syntactic techniques, to gain some understanding of how these two filtering techniques can be combined successfully. It is possible that more sophisticated filterbots would have performed much better than the simple filterbots we studied. However, the results are important in showing that even simple filterbots add value to user ratings.

Our results were based on a collaborative filtering dataset from the GroupLens Research public trial. While this trial is still one of the largest trials conducted on streams of discussion data, the ratings density in the data is very low. Newsgroups such as rec.humor would require hundreds or thousands of users to achieve an average of even ten ratings per article, in part because the newsgroup has so many unfunny articles that there is substantial incentive to skip any article that doesn't have a strong recommendation.

Possible consequences of low rating density include:

- Less personalization within the recommendation process, since there are too few ratings for the algorithm to be "fussy" about matches.
- Lower accuracy and coverage in the "no filterbot" case than would be the case otherwise.

At the same time, low rating density is a real-world condition that presents the problem that filterbots are intended to solve.A related limitation is the small number of users studied. We had 76 users who rated articles that overlapped the filterbot ratings. Of these users, many rated only a few articles and therefore contributed little to the analysis. Even though this study should be replicated with a larger user set, we believe it reflects the largest study of its type, and therefore can serve as a basis for additional experimentation.

Finally, we recognize that Usenet News is, in general, a high-noise information stream. We selected two moderated newsgroups to ameliorate that effect, but should caution those trying to generalize the work to low-noise environments that very simple filterbots may not add enough value to be useful.

### Applications of this Work

There are several interesting applications of our architecture and results. A number of real-world collaborative filtering systems recommend objects from immense sets (e.g., books in print, web pages) where filterbots could help address ratings sparsity.

A particularly exciting idea is the use of the filterbot framework as a mechanism for increasing the utility of agent software. Few agents today are sufficiently powerful and general to merit individual use, so integrating them into a framework with collaborative filtering and other agents can help them reach the threshold of utility. Also, individual filtering agents aren't inherently god or bad; they are more useful to some users and less useful to others. Integrating them into a collaborative filtering framework helps match users to agents. It also helps address the case where a particular agent has no information to communicate—a feature that may have helped our filterbots.

### Future Work

Our results represent only a first step in understanding the ways in which content filtering can be successfully

integrated into collaborative filtering. Among the issues we would like to study in the future are:

- the interaction of sets of filterbots in the same system.
- the process of selecting proper filterbots for an application domain; we clearly could not know in advance which algorithms would work for the newsgroups, and indeed certain cases resulted in a drop in overall accuracy.
- the use of more complex filterbot algorithms, including algorithms that learn.
- a "personal filterbot" system where each user has "agent filterbots" attempting to learn her tastes.
- the value of filterbots for users with few ratings. Should the engine only phase in filterbots after users have a certain number of ratings and established correlations?

In addition to these questions, we have a large number of particular filterbots and applications that we'd like to explore, including filterbots for movies and other non-textual media.

## CONCLUSIONS

This paper makes three contributions to the field of collaborative filtering.

First, it defines and implements a model for integrating content-based ratings into a collaborative filtering system. This filterbot model allows collaborative filtering systems to address sparsity and early-rater problems by tapping the strength of content filtering techniques. Second, it identifies and evaluates metrics for assessing the effectiveness of filterbots specifically, and filtering system enhancements in general. Third, it experimentally validates the filterbot approach by showing that even simple filterbots such as spell checking can increase the utility for users of sparsely populated collaborative filtering systems.

## ACKNOWLEDGEMENTS

The authors gratefully acknowledge the contributions of all the members of the GroupLens Research team, and the support of Net Perceptions Inc. and the National Science Foundation under grant IRI-9613960. We also thank the anonymous reviewers for their valuable comments.

## REFERENCES

1. Avery, C. and Zeckhauser, R. Recommender Systems for Evaluating Computer Messages. *CACM*. 40(3), pp. 88-89, March 1997.
2. Balabanovic, M. and Shoham, Y. Fab: Content-Based, Collaborative Recommendation. *CACM*. 40(3), pp. 66-72, March 1997.
3. Belkin, N. J. and Croft, B. W. Information Filtering and Information Retrieval: Two Sides of the Same Coin? *CACM*. 35(2), December 1992.
4. Goldberg, D., Nichols, D., Oki, B. M. and Terry, D. Using Collaborative Filtering to Weave an Information Tapestry. *CACM*. Dec. 1992.
5. Hill, W., Stead, L., Rosenstein, M., Furnas, G. Recommending and Evaluating Choices in a Virtual Community of Use. *Proceedings of CHI '95*.
6. Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R. and Riedl, J. GroupLens: Applying Collaborative Filtering to Usenet News. *CACM*. 40(3), March 1997.
7. Le, C. T. and Lindgren, B. R. Construction and Comparison of Two Receiver Operating Characteristics Curves Derived from the Same Samples. Biom. J. 37(7), pp. 869-877, July 1995.
8. Maes, P. Agents that Reduce Work and Information Overload. *CACM*. July 1994.
9. Maltz, D. and Ehrlich, K. Pointing the Way: Active Collaborative Filtering. *Proceedings of CHI '95*.
10. Miller, B., Riedl, J. and Konstan, J. Experiences with GroupLens: Making Usenet Useful Again. *Proceedings of the 1997 Usenix Technical Conference*.
11. Moukas, A. and Zacharia, G. Evolving a Multi-agent Information Filtering Solution in Amalthaea. In *Proceedings of Autonomous Agents 97*.
12. Resnick, P. and Varian, H. R. Recommender Systems. *CACM*. 40(3), pp. 56-58, March 1997.
13. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. *Proceedings of CSCW '94*. Chapel Hill, NC. 1994.
14. Rucker, J. and Polano, M. J. Siteseer: Personalized Navigation for the Web. *CACM*. 40(3), March 1997.
15. Salton, G. and McGill M. J. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc. 1983.
16. Shardanand, U. and Maes, P. Social Information Filtering: Algorithms for Automating "Word of Mouth". In *Proceedings of the CHI '95*. Denver, CO. May 1995.
17. Terveen, L., Hill, W., Amento, B., McDonald, D. and Creter, J. PHOAKS: A System for Sharing Recommendations. *CACM*. 40(3), pp. 59-62, March 1997.