Combining Collaborative Filtering with Personal Agents for Better Recommendations

Nathaniel Good, J. Ben Schafer, Joseph A. Konstan, Al Borchers, Badrul Sarwar, Jon Herlocker, and John Riedl

GroupLens Research Project
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455
http://www.grouplens.org/

Abstract

Information filtering agents and collaborative filtering both attempt to alleviate information overload by identifying which items a user will find worthwhile. Information filtering (IF) focuses on the analysis of item content and the development of a personal user interest profile. Collaborative filtering (CF) focuses on identification of other users with similar tastes and the use of their opinions to recommend items. Each technique has advantages and limitations that suggest that the two could be beneficially combined.

This paper shows that a CF framework can be used to combine personal IF agents and the opinions of a community of users to produce better recommendations than either agents or users can produce alone. It also shows that using CF to create a personal combination of a set of agents produces better results than either individual agents or other combination mechanisms. One key implication of these results is that users can avoid having to select among agents; they can use them all and let the CF framework select the best ones for them.

Introduction

Recommender systems help individuals and communities address the challenges of information overload. Information filtering recommenders look at the syntactic and semantic content of items to determine which are likely to be of interest or value to a user. Collaborative filtering recommenders use the opinions of other users to predict the value of items for each user in the community. For example, in the domain of movie selection, content filtering would allow recommendation based on the movie genre (horror, comedy, romance, etc.) and cast/credits (Woody Allen, Steven Spielberg, Bette Midler). Collaborative filtering, by contrast, might be completely unaware of genre and cast, but would know that a group of like-minded people recommends "Hoop Dreams" and suggests avoiding "Dumb and Dumber."

In this work, we examine collaborative filtering, personal information filtering agents, and mechanisms for combining them to produce a better recommender system. The next section reviews existing approaches to alleviating information overload, including a variety of content-based and collaborative approaches, and presents our model for how these approaches can be more effective when combined. The following sections present our experimental design and results. We conclude with observations about the implications of these results.

Information Overload: Problem and Approaches

Each day, more and more books, journal articles, web pages, and movies are created. As each new piece of information competes for our attention, we quickly become overwhelmed and seek assistance in identifying the most interesting, worthwhile, valuable, or entertaining items on which we should expend our scarce money and time. Historically, humans have adapted well to gluts of information. Our senses are tuned to notice change and the unusual. Our ability to communicate allows us to collaboratively address large problems. And, we have developed an astonishingly good ability to make quick judgements—indeed, we often *can* judge a book by its cover, an article by its title or abstract, or a movie by its trailer or advertisement. Today we are also finding that it is becoming easier and easier to produce and publish content. As computers, communication, and the Internet make it easier for anyone and everyone to speak to a large audience, we find that even our well-developed filtering skills may be inadequate.

In response to the challenge of information overload, we have sought to develop useful recommender systems—systems that people can use to quickly identify content that will likely interest them. This project draws from work in creating recommender systems for movies – film fans tell the MovieLens system (movielens.umn.edu) how much they like or dislike movies they've already seen, and MovieLens recommends other movies they would likely enjoy.

There are three different technologies that are commonly used to address information overload challenges. Each technology focuses primarily on a particular set of tasks or questions. *Information retrieval* focuses on tasks involving fulfilling ephemeral interest queries such as finding the movies directed by Woody Allen. *Information filtering* focuses on tasks involving classifying streams of new content into categories, such as finding any newly released movies directed by Steven Spielberg (to consider watching) or any newly released movies without an English-language soundtrack or subtitles (to reject). *Collaborative filtering* focuses on answering two questions:

- *Which items (overall or from a set) should I view?*
- *How much will I like these particular items?*

Each of these technologies has a role in producing an effective recommender system.

Information retrieval (IR) systems focus on allowing users to express queries to select items that match a topic of interest to fulfill a particular information need. They may index a collection of documents using either the full text of the document or document abstracts. For non-textual items such as movies, IR systems index genres, keywords, actors, directors, etc. IR systems are generally optimized for ephemeral interest queries, such as looking up a topic in the library. (Belkin and Croft 1992) Internet search engines are popular IR systems, and the Internet Movie Database (www.imdb.com) provides extensive support for IR queries on movies.

An IR front-end is useful in a recommender system both as a mechanism for users to identify specific movies about which they would like to express an opinion and for narrowing the scope of recommendation. For example, MovieLens allows users to specifically request recommendations for newer movies, for movies released in particular time periods, for particular movie genres such as comedy and documentary, and for various combinations of movie. Information retrieval techniques are less valuable in the actual recommendation process, since they capture no information about user preferences other than the specific query. For that reason, we do not consider IR further in this paper.

Information filtering (IF) systems require a profile of user needs or preferences. The simplest systems require the user to create this profile manually or with limited assistance. Examples of these systems include: “kill files” that are used to filter out advertising, e-mail filtering software that sorts e-mail into categories based on the sender, and new-product notification services that request notification when a new book or album by a favorite author or artist is released. More advanced IF systems may build a profile by learning the user's preferences. A wide range of agents including Maes' agents for e-mail and Usenet news filtering (Maes 1995) and Lieberman's Letizia (Lieberman 1997) employ learning techniques to classify, dispose of, or recommend documents based on the user's prior actions. Similarly, Cohen's Ripper system has been used to classify e-mail (Cohen 1996); alternative approaches use other learning techniques and term frequency (Boone 1998)

Information filtering techniques have a central role in recommender systems. IF techniques build a profile of user preferences that is particularly valuable when a user encounters new content that has not been rated before. An avid Woody Allen fan doesn't need to wait for reviews to decide to see a new Woody Allen film, and a person who hates horror films can as quickly dismiss a new horror film without regret. IF techniques also have an important property that they do not depend on having other users in the system, let alone users with similar tastes. IF techniques can be effective, as we shall see, but they suffer certain drawbacks, including requiring a source of content information, and not providing much in the way of serendipitous discovery; indeed, a Woody Allen-seeking agent would likely never discover a non-Woody Allen drama that just happens to appeal greatly to most Woody Allen fans.

Collaborative filtering (CF) systems build a database of user opinions of available items. They use the database to find users whose opinions are similar (i.e., those that are highly correlated) and make predictions of user opinion on an item by combining the opinions of other like-minded individuals. For example, if Sue and Jerry have liked many of the same movies, and Sue liked *Titanic*, which Jerry hasn't seen yet, then the system may recommend *Titanic* to Jerry. While Tapestry (Goldberg et al. 1992), the earliest CF system, required explicit user action to retrieve and evaluate ratings, automatic CF systems such as GroupLens (Resnick et al. 1994) (Konstan et al. 1997) provide predictions with little or no user effort. Later systems such as Ringo (Shardanand and Maes 1995) and Bellcore's Video Recommender (Hill et al. 1995) became widely used sources of advice on music and movies respectively. More recently, a number of systems have begun to use observational ratings; the system infers user preferences from actions rather than requiring the user to explicitly rate an item (Terveen et al. 1997). In the past year, a wide range of web sites have begun to use CF recommendations in a diverse set of domains including books, grocery products, art, entertainment, and information.

Collaborative filtering techniques can be an important part of a recommender system. One key advantage of CF is that it *does not* consider the content of the items being recommended. Rather than map users to items through "content attributes" or "demographics," CF treats each item and user individually. Accordingly, it becomes possible to discover new items of interest simply because other people liked them; it is also easier to provide good recommendations even when the attributes of greatest interest to users are unknown or hidden. For example, many movie viewers may not want to see a particular actor or genre so much as "a movie that makes me feel good" or "a smart, funny movie." At the same time, CF's dependence on human ratings can be a significant drawback. For a CF system to work well, several users must evaluate each item; even then, new items cannot be recommended until some users have taken the time to evaluate them. These limitations, often referred to as the *sparsity* and *first-rater problems*, cause trouble for users seeking obscure movies (since nobody may have rated them) or advice on movies about to be released (since nobody has had a chance to evaluate them).

Hybrid Recommender Systems

Several systems have tried to combine information filtering and collaborative filtering techniques in an effort to overcome the limitations of each. Fab (Balabanovic and Shoham 1997) maintains user profiles of interest in web pages using information filtering techniques, but uses collaborative filtering techniques to identify profiles with similar tastes. It then can recommend documents across user profiles. (Basu, Hirsh, and Cohen 1998) trained the Ripper machine learning system with a combination of content data and training data in an effort to produce better recommendations. Researchers working in collaborative filtering have proposed techniques for using IF profiles as a fall-back, e.g., by requesting predictions for a director or actor when there is no information on the specific movie, or by having dual systems and using the IF profile when the CF system cannot produce a high-quality recommendation.

In earlier work, Sarwar, et al. (1998) showed that a simple but consistent rating agent, such as one that assesses the quality of spelling in a Usenet news article, could be a valuable participant in a collaborative filtering community. In that work, they showed how these *filterbots*—ratings robots that participate as members of a collaborative filtering system – helped users who agreed with them by providing more ratings upon which recommendations could be made. For users who did not agree with the filterbot, the CF framework would notice a low preference correlation and not make use of its ratings.

This work extends the filterbot concept in three key ways. First, we use a more intelligent set of filterbots, including learning agents that are personalized to an individual user. Second, we apply this work to small communities, including using CF to serve a single human user. Third, we evaluate the simultaneous use of multiple filterbots. In addition, we explore other combination mechanisms as alternatives to CF. We demonstrate that CF is a useful framework both for integrating agents and for combining agents and humans.

Hypotheses and Experimental Design

In this paper, we systematically explore the value of collaborative filtering, information filtering, and different combinations of these techniques for creating an effective personal recommendation system. Specifically, we look at four key models as shown in figure 1:

- Pure collaborative filtering using the opinions of other community members
- A single personalized "agent" – a machine learning or syntactic filter
- A combination of many "agents"
- A combination of multiple agents and community member opinions

The experimental design uses two tiers. First, where there are several implementations for a particular model, we evaluate them to find the model that provides the best filtering. Second, we compare the best implementation

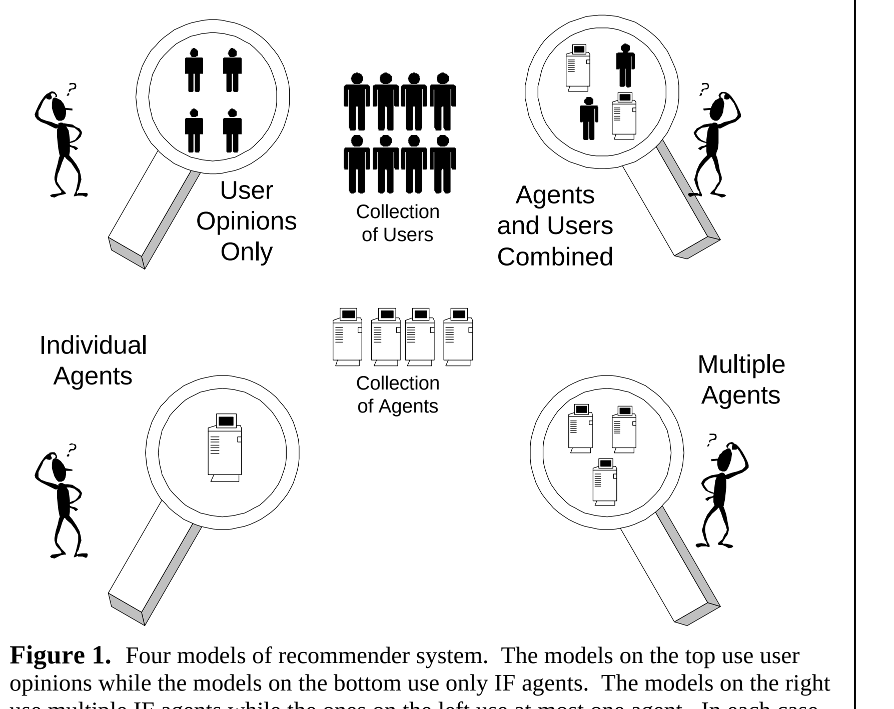

Figure 1. Four models of recommender system. The models on the top use user opinions while the models on the bottom use only IF agents. The models on the right use multiple IF agents while the ones on the left use at most one agent. In each case, the user receives personal recommendations.

from each model with the other implementations. These are operationalized as four primary hypotheses below.

H1. The opinions of a community of users provide better recommendations than a single personalized agent.

H2. A personalized combination of several agents provides better recommendations than a single personalized agent.

H3. The opinions of a community of users provides better recommendations than a personalized combination of several agents.

H4. A personalized combination of several agents and community opinions provides better recommendations than either agents or user opinions alone.

The context in which these hypotheses are tested is a small, anonymous community of movie fans. The combination of small size and non-textual content cause disadvantages for both collaborative filtering and information filtering; it provides a middle-ground between the common contexts for collaborative filtering (many users, little content information) and information filtering (one user, much content information).

Data Set

The user ratings for this experiment were drawn from the MovieLens system (http://movielens.umn.edu) which has more than 3 million ratings from over 80,000 users. Fifty users were selected at random from the set of users with more than 120 movie ratings. For each user, three sets of movies/ratings were selected at random without replacement. The first set of 50 ratings, termed the **training set**, was set aside for use in training the personalized information filtering agents. The second set of 50 ratings, termed the **correlation set** was used when combining users, agents, or both together. The final set of 20 ratings served as the **test set**. In each experiment, the test ratings of the target user were withheld and compared against the recommendation value produced by the system.

Metrics

Recommender systems researchers use several different measures for the quality of recommendations produced.

Coverage metrics evaluate the number of items for which the system could provide recommendations. In many systems, coverage decreases as a function of accuracy—the system can produce fewer accurate recommendations or more inaccurate ones. Because our information filtering systems provide total coverage, we do not report coverage except as part of the analysis of the standard CF system.

Statistical accuracy metrics evaluate the accuracy of a filtering system by comparing the numerical prediction values against user ratings for the items that have both predictions and ratings. (Shardanand and Maes, 1995) and (Sarwar et al, 1998) have both used mean absolute error (MAE) to measure the performance of a prediction engine. Other metrics used include root mean squared error (Sarwar et al. 1998) and correlation between ratings and predictions (Hill et al. 1995) (Konstan et al. 1997) (Sarwar et al. 1998). Our experience has shown that these metrics typically track each other closely. We have chosen to report mean absolute error, therefore, because it is the most commonly used and the easiest to interpret directly.

Decision-support accuracy metrics evaluate how effective a prediction engine is at helping a user select high-quality items from the item set. These metrics are based on the observation that, for the majority of users, filtering is a binary operation – they will either view the item, or they will not. If this is true, then whether an item has a rating of 1.5 or 2.5 on a five-point scale is irrelevant if the user only views items with a rating of 4 or higher. The most common decision-support accuracy measures are reversal rate, weighted errors, and ROC sensitivity. *Reversal rate* is the frequency with which the system makes recommendations that are extremely wrong. On a five point scale, it is commonly defined as the percentage of recommendations where the recommendation was off by 3 points or more. Weighted error metrics give extra weight to large errors that occur when the user has a strong opinion about the item. For example, errors might count double or more when the user considers the item a favorite (5 out of 5). *ROC sensitivity* is a signal processing measure of the decision making power of a filtering system. Operationally, it is the area under the receiver operating characteristic curve (ROC) – a curve that plots the *sensitivity* vs. *1 - specificity* of the test (Swets 1988). 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. Points on the ROC curve represent trade-offs supported by the filter. A good filter might allow the user to choose between receiving 90% of the good items while accepting only 10% of the bad ones, or receiving 95% of the good ones with 20% of the bad ones. A random filter always accepts the same percentage of the good and the bad items. The ROC sensitivity ranges from 0 to 1 where 1 is perfect and 0.5 is random.

We use ROC sensitivity as our decision support accuracy measure. To operationalize ROC, we must determine which items are "good" and which are "bad." We use the user's own rating, with a mapping that 4 and 5 are good and 1,2, and 3 are bad. Our experience has shown that this reflects user behavior on MovieLens. We found that one user had no movies rated below 4; we eliminated that user from the statistics compiled for each experiment.

Evaluating the hypotheses in the face of multiple metrics can be a challenge. We considered it important to consider both statistical and decision-support accuracy in evaluating different recommender systems. When several agents, for example, provide different but incomparable trade-offs among the two metrics, we consider each one to be a possible "best agent" and compare each of them against the alternative recommender. We consider one alternative to dominate another, however, if there is a significant improvement in one metric and no significant difference in the other.

Statistical significance is assessed for mean absolute errors using the Wilcoxan test on paired errors. Differences reported as significant are based on a significance level of $p<0.05$. Statistical significance assessment for ROC sensitivity is less clear;[*] from experience we therefore assert that changes of 0.01 or more are "meaningful" and smaller differences are "not meaningful."

Experimental Components

Our hypotheses are based on four models of recommender system:

- user opinions only,
- individual IF agents,
- combinations of IF agents, and
- combinations of IF agents and user opinions.

In this section, we describe the variety of implementations of these models with an overview of how we constructed each implementation. The effectiveness of each implementation is reported in the results section.

User Opinions Only. Extensive research has already been performed on the problem of generating recommendations from a set of user opinions. Nearest-neighbor collaborative filtering is already generally accepted to be the most effective mechanism for performing this task, and we therefore use it (Breese, 1998). In particular, we use the DBLens research collaborative filtering engine developed by the GroupLens Research project for exploration of collaborative filtering algorithms. DBLens allows experimenters to control several parameters that trade among performance, coverage, and accuracy. For our experiments, we set each of these to prefer maximum coverage and to use all data regardless of performance.

The CF result set was computed for each user by loading the correlation data set (50 ratings per user) into the engine, then loading the test set (20 ratings per user) for each user, and requesting a prediction for each test set movie for each user. DBLens has a control that allows us to ignore a user's rating when making a prediction for that user. The resulting 20 predictions per user were compared against that user's ratings to produce error and ROC statistics.

[*]There are statistical measures to compare ROC curves themselves, but the measures that we have found for comparing areas under the curve appear to overstate statistical significance.

Individual IF Agents. Three types of IF agents, or filterbots, were created and studied in this project: DGBots, RipperBot, and a set of GenreBots.

Doppelganger Bots (DGBots) are personalized bots that create profiles of user preferences and generate predictions using IR/IF techniques (specifically, a modified TFIDF, (Salton and Buckley 1987) based upon the content features of each movie. We created three DGBots, one that used only cast data, one that used only descriptive keywords, and one that used both. These data were found at the Internet Movie Database (http://www.imdb.com/). Each DGBot was implemented similarly, so we will describe the keyword DGBot here.

To produce personal recommendations for movies, the keyword DGBot followed five steps:

1. Create an IDF vector that represents the relative scarcity of each keyword in the movie set.
2. Create a term frequency vector for each movie indicating which keywords occur.
3. Build a user profile of weights associated with each term
4. Produce a score for each movie based on the user weights.
5. Rank order the movies and divide into recommendation bands.

The IDF vector is created using the following formula for the value associated with each keyword:

$$idf = \log_2\left(\frac{N}{O}\right)$$

N is the total number of movies and O is the number of movies for which that keyword is used.

We modified traditional TFIDF by counting each keyword as either occurring (1) or not occurring (0) in any given movie. Accordingly, the TF vector for a movie is produced by inserting a 1 for each keyword and 0 elsewhere.

Building the user profile requires a balanced set of user ratings, so we subtract 3 from each rating to transform them to a -2 to +2 scale. For each movie in the 50-rating training set, we produce a keyword preference vector that is the product of the transformed rating, the movie's TF vector, and the IDF vector. We then normalize the keyword preference vector to length 1. The mean of the user's 50 keyword preference vectors is the user profile.

The DGBot produces ratings for all movies at once. For each movie, it computes the dot product of the user profile vector and the TF vector. Those scores are then ranked and broken into rating levels with a distribution matching the MovieLens overall rating distribution. The top 21% of movies received a rating of 5, the next 34% a rating of 4, the next 28% a 3, the next 12% a 2, and the bottom 5% a 1. While each user has a separate user profile vector and set of recommendations, the TF and IDF vectors could be re-used from user to user.

RipperBot was created using Ripper, an inductive logic program created by William Cohen (Cohen, 1995). We found that Ripper performed best when trained on a set of data limited to genre identifiers and the 200 most frequent keywords. Ripper also works best when asked to make binary decisions, so for each user we trained four Ripper instances, tuned to distinguish between 5/4321, 54/321, 543/21, and 5432/1 respectively. Each instance was trained on the 50-rating training set along with the identifiers and keywords for those 50 movies. After training, we asked each instance to classify the entire set of movies and summed the number of Ripper instances that indicated the higher value and added one to create a recommendation value.

Ripper requires substantial tuning; we experimented with several parameters and also relied on advice from (Basu Hirsh and Cohen, 1998). In particular, we adjusted default settings to *allow negative tests in set value attributes* and experimented by varying the *loss ratio*. We found a loss ration of 1.9 to give us the best results.

The GenreBots consisted of 19 simple bots that rated each movie a 5 if the movie matched the bot's genre and a 3 otherwise. For example, *Toy Story*, which is a children's animated comedy would receive a 5 from the ChildrensBot, the AnimatedBot and the ComedyBot, and a 3 from each of the remaining bots. Genre data was obtained from IMDB.

A Mega-GenreBot was created for each user. This was done by using linear regression and training the bot on each user's training set. A user's known rating was treated as a dependent variable of the 19 individual GenreBots. The regression coefficients formed an equation that could then be used to generate predictions for each other movie from the genre identifiers.

Combinations of IF Agents. We identified four different strategies for combining agents: selecting one agent for each person, averaging the agents together, using regression to create a personal combination, and using CF to create a personal combination. For all but the first of these, we found it valuable to create two combinations: one that used all 19 GenreBots and one that used the Mega-GenreBot. Adding the 3 DGBots and RipperBot, we refer to these as 23-agent and 5-agent versions, respectively.

BestBot. The best agent per user was selected by testing each bot on the correlation data set (50 ratings) and selecting the bot with the lowest MAE. BestBot then used the ratings generated by that bot for the test data set to produce statistics for evaluation.

Agent Average. The average combination was produced by taking the arithmetic mean of the 5 or 23 agent recommendations, respectively.

Regression. We used linear regression to produce a "best fit" combination for a given user. To do this we used the predictions on the correlation sets for the 23 and 5 agents respectively as the independent variables and the known user's rating as the dependent variable. Using the resultant weights, we could generate predictions for the movies in the test sets by creating linear combinations of the agents' recommendations.

CF Combination. We used the DBLens CF engine to create a CF combination of agents. For this purpose, we loaded all ratings from the 5 or 23 agents into the engine, along with the user's 50 ratings from the correlation set. We generated predictions for the user's 20 test movies. The ratings database was cleared after each user. The parameters used were the same as for the simple CF case.

Combination of Users and IF Agents. Because user ratings were incomplete, and because CF with 23 agents proved to be the most effective combination of IF agents, we used CF to combine the 23 agents and all 50 users. The method is identical to the CF combination of agents except that we also loaded the ratings for the other 49 users. Again, the database was cleared after each user.

Results

✗ H1: Collaborative Filtering better than Single Agents

We hypothesized that collaborative filtering using the opinions of the 50-user community would provide better results than any individual agent. To compare these, we first identified the best individual agent. We evaluated the three DGBots, RipperBot, the 19 individual genreBots, and the personalized Mega-GenreBot (see table 1). Of these, only RipperBot, Mega-GenreBot, and the DGBot that used both cast and keywords were not dominated by other agents. RipperBot had the highest accuracy (lowest MAE) by far, but low ROC sensitivity (poor decision support). The combined DGBot has the

Bot or Method	MAE	ROC	Bot or Method	MAE	ROC
ActionBot	1.0755	0.4925	RomanceBot	1.0897	0.4931
AdventureBot	1.0653	0.5148	Sci-FiBot	1.0714	0.5026
AnimationBot	1.0612	0.5017	ThrillerBot	1.0897	0.4815
ChildrensBot	1.0346	0.5155	UnknownBot	1.0816	0.4922
ComedyBot	1.2652	0.4767	WarBot	1.0428	0.5187
CrimeBot	1.1000	0.5006	WesternBot	1.0673	0.5078
DocumntryBot	1.0918	0.4927	DGBot Cast	1.2775	0.5673
DramaBot	1.0591	0.5151	DGBot Kwd	1.1397	0.5706
FamilyBot	1.0489	0.5161	DGBot Comb.	1.1428	0.5771
Film-NoirBot	1.0959	0.4924	MegaGnrBot	0.9578	0.5742
HorrorBot	1.0632	0.5066	RipperBot	0.8336	0.5236
MusicalBot	1.0673	0.5182	CF of Users	0.9354	0.5788
MysteryBot	1.0734	0.5114			

Table 1. Individual Bots vs. CF of Users

highest ROC sensitivity, but relatively low accuracy. The Mega-GenreBot has the second-best accuracy and second-best decision-support. We compare these three against the results of collaborative filtering using user opinions.

Collaborative filtering is significantly less accurate than RipperBot, but has a meaningfully higher ROC sensitivity value. In effect, while RipperBot avoids making large errors, it performs little better than random at helping people find good movies and avoid bad ones. If accuracy were paramount, H1 would be rejected.

Collaborative filtering is significantly more accurate than the combined DGBot and has comparable ROC sensitivity. While both approaches provide comparable support for decision-making, on average the DGBot is more than 20% less accurate. If decision-support were paramount, H1 would be accepted.

The Mega-GenreBot is slightly worse than collaborative filtering of user opinions on both MAE and ROC, but the differences were not statistically significant. We would consider Mega-GenreBot to be a good pragmatic substitute for user-based collaborative filtering for a small community. Furthermore, the collaborative filtering result was only able to provide coverage of 83% (other desired recommendations could not be made due to a lack of ratings for those movies).

Accordingly, overall we reject H1. A more accurate alternative exists (RipperBot), and comparably accurate and valuable alternatives exist without the problem of reduced coverage.

✓ H2: Many Agents better than Just One

We hypothesized that combining several agents would yield better results than any single personalized agent. In testing H1, we found that for single agents, RipperBot had the best accuracy value (MAE), DGBot Combo had the best decision support value (ROC) and the Mega-GenreBot was competitive with both values. In table 2 we compare these values to those obtained from the seven methods of combining the agents – regression, agent average, collaborative filtering of a single user and its bots, and manually selecting the "best bot."

Bot or Method	MAE	ROC	Bot or Method	MAE	ROC
5 agent regress	0.8610	0.6030	RipperBot	0.8336	0.5236
23 agent regress	0.9729	0.5676	MegaGenreBot	0.9578	0.5742
5 agent avg.	0.8990	0.6114	DG Combo Bot	1.1428	0.5771
23 agent avg.	0.9579	0.5760	CF 1 usr, 5 bots	0.9682	0.6071
Best Bot	0.8714	0.5173	CF 1 usr, 23 bts	0.8343	0.6118

Table 2. Individual Bots vs. Combined Bots

Collaborative filtering using all 23 (CF23) agents is clearly the best combination method, with a significant accuracy advantage over all of the other combinations and similar or better ROC values. CF23 provides both MAE and ROC advantages over both the Mega-GenreBot and the DGBot Combo.

The remaining interesting comparison is between CF23 and RipperBot. We conclude that CF23 is better because there was no significant difference in MAE, and the ROC value for CF23 was dramatically better than for RipperBot. Accordingly we accept H2. We also observe that H2 depended on using collaborative filtering technology; no other combination method was close to dominating RipperBot's accuracy.

✗ H3: CF of Users better than Combination of Agents

At this stage, it is clear that we must reject H3. Table 3 summarizes the results, but we recognize that collaborative filtering with a group of 50 users is indeed not as accurate or valuable as we had hypothesized.

Bot or Method	MAE	ROC	Bot or Method	MAE	ROC
CF (users only)	0.9354	0.5788	CF 1 usr, 23 bts	0.8343	0.6118

Table 3. CF of Users vs. Combined Bots

✓ H4: Agents and Users Together is best overall

We hypothesized that the combination of the opinions of a community of users and the personalized agents for a given user will provide that user with better results than either users alone or agents alone. From both H2 and H3 we found that collaborative filtering of a single user and that user's 23 agents provides the best accuracy and decision support of all agent-only or user-only methods tested. Table 4 shows a small, but statistically significant improvement in accuracy resulting from including the other users in the collaborative filtering mix. ROC also improves, but not by a meaningful amount.

Bot or Method	MAE	ROC	Bot or Method	MAE	ROC
CF 50 usr, 23 bt	0.8303	0.6168	CF 1 usr, 23 bts	0.8343	0.6118

Table 4. CF of Users and Bots vs. Combined Bots

Accordingly, we accept H4 and find that a mixed collaborative filtering solution that uses users and agents does indeed provide the best overall results.

Discussion

The most important results we found were the value of combining agents with CF and of combining agents and users with CF. In essence, these results suggest that an effective mechanism for producing high-quality recommendations is to throw in any available data and allow the CF engine to sort out which information is useful to each user. In effect, it becomes less important to invent a brilliant agent, instead we can simply invent a collection of useful ones. We should point out that these experiments tested the quality of the resulting recommender system, not the performance or economics of such a system. Current CF recommendation engines cannot efficiently handle "users" who rate all items and re-rate them frequently as they "learn." To take advantage of learning agents, these engines must be redesigned to accommodate "users" with dynamic rating habits. We are examining several different CF engine designs that could efficiently use filterbots.

We were also pleased, though somewhat surprised, to find that CF outperformed linear regression as a combining mechanism for agents. While linear regression should provide an optimal linear fit, it appears that CF's non-optimal mechanism actually does a better job avoiding overfitting the data when the number of columns approaches the number of rows. CF also has the advantage of functioning on incomplete (and indeed very sparse) data sets, suggesting that it retains its value as a useful combination tool whenever human or agents are unlikely to rate each item.

We were surprised by several of the results that we'd found, and sought to explain them. Foremost, we clearly overestimated the value of collaborative filtering for a small community of 50 users. In retrospect, our expectations may have been built from our own positive experiences when starting CF systems with a small group of researchers and friends. Those successes may have been due in part to close ties among the users; we often had seen the same movies and many had similar tastes. Using real users resulted in real diversity which may explain the lower, and more realistic, value. Future work should both incorporate larger user sets (other experiments have consistently shown MAE values in the range of 0.71-0.73 and ROC sensitivity values near 0.72 for MovieLens communities with thousands of users) and look explicitly at closer-knit communities to see whether a smaller but more homogeneous community would have greater benefits from collaborative filtering.

We also were surprised by the results we achieved using Ripper. We were impressed by its accuracy, after extensive tuning, but dismayed by how close to random it was in distinguishing good from bad movies. We are still uncertain as to why RipperBot performs as it does, and believe further work is needed to understand why it behaves as it does and whether it would be possible to train it to perform differently.

In the future, we plan to examine further combinations of users and agents in recommender systems. In particular, we are interested in developing a combined community where large numbers of users and agents co-exist. One question we hope to answer is whether users who agree with each other would also benefit from the opinions of each other's trained agents.

Acknowledgements

Funding for this research was provided by the National Science Foundation under grants IIS 9613960, IIS 9734442, and DGE 9554517. Support was also provided by Net Perceptions Inc. We would also like to thank members of the GroupLens research team, especially Hannu Huhdanpaa for developing the initial testbed.

References

Balabanovic, M., and Shoham, Y. 1997. Fab: Content-Based, Collaborative Recommendation. *Communications of the ACM* 40(3):66-72.

Basu C., Hirsh H., and Cohen, W.W. 1998. Using Social and Content-Based Information in Recommendation. In *Proceedings of the AAAI-98,*:AAAI Press.

Belkin, N., and Croft, B.W. 1992. Information Filtering and Information Retrieval: Two Sides of the Same Coin?. *Communications of the ACM* 35(12):29-38.

Boone, G. 1998. Concept Features in Re:Agent, an Intelligent Email Agent. In *The Second International Conference on Autonomous Agents*, 141-148, Minneapolis/St. Paul, MN:ACM.

Breese, J. Heckerman, D., and Kadie, C. 1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, Madison, WI.

Cohen, W.W. 1996. Learning Rules that Classify E-mail. In *Proceeding of the AAAI Spring Symposium on Machine Learning in Information Access,*: AAAI Press.

Cohen, W.W. 1995. Fast Effective Rule Induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, Lake Tahoe, CA.:AAAI Press.

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

Hill, W., Stead, L., Rosenstein, M., and Furnas, G., 1995. Recommending and Evaluating Choices in a Virtual Community of Use. In *Proceedings of ACM CHI'95*, 194-201. Denver, CO.: ACM.

Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R., and Riedl, J. 1997Applying Collaborative Filtering to Usenet News. *Communications of the ACM* 40(3):77-87.

Lieberman, H., 1997. Autonomous Interface Agents. In *Proceedings of ACM CHI 97*,67-74,:ACM

Maes, P. 1995. Agents that Reduce Work and Information Overload. In *Readings in Human-Computer Interaction, Toward the Year 2000*, :Morgan Kauffman.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J. 1994. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In Proceedings of 1994 Conference on Computer Supported Collaborative Work, 175-186.: ACM.

Salton, G., Buckley, C. 1987. Term Weighting Approaches in Automatic Text Retrieval., Technical Report, Dept. of Computer Science, Cornell Univ.

Sarwar, B.M., Konstan, J.A., Borchers, A., Herlocker, J.L., Miller, B.N., and Riedl, J. 1998. Using Filtering Agents to Improve Prediction Quality in the Grouplens Research Collaborative Filtering System. In *Proceedings of CSCW '98*, Seattle, WA.: ACM.

Shardanand, U., and Maes, P. 1995. Social Information Filtering: Algorithms for Automating "Word of Mouth". In *Proceedings of ACM CHI '95*,. Denver, CO.: ACM.

Swets, J.A. 1988. Measuring the Accuracy of Diagnostic Systems. *Science* 240:1285-1289.

Terveen, L., Hill, W., Amento, B., McDonald, D., Creter J. 1997. PHOAKS: A System for Sharing Recommendations. Communications of the ACM 40(3):59-62.