An Algorithmic Framework for Performing Collaborative Filtering

Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl
{herlocke,konstan,borchers,riedl}@cs.umn.edu
Dept. of Computer Science and Engineering
University of Minnesota
www.cs.umn.edu/Research/GroupLens/

Abstract Automated collaborative filtering is quickly becoming a popular technique for reducing information overload, often as a technique to complement content-based information filtering systems. In this paper we present an algorithmic framework for performing collaborative filtering and new algorithmic elements that increase the accuracy of collaborative prediction algorithms. We then present a set of recommendations on selection of the right collaborative filtering algorithmic components.

1 Introduction

Automated collaborative filtering is quickly becoming a popular technique for reducing information overload, often as a technique to complement content-based information filtering systems. Automated collaborative filtering has seen considerable success on the Internet, being used at sites like Amazon.com—the largest book store on the Internet, CDNow.com—the largest CD store on the Web, and MovieFinder.com—one of the most visited movie sites on the Internet.

Content-based and collaborative filtering use different types of data to arrive at a filtering decision. Content-filtering tools select the right information for the right people by comparing representations of content contained in the documents to representations of content that the user is interested in. Content-based information filtering has proven to be effective in locating textual documents relevant to a topic using techniques such as vector-space queries[19], "intelligent" agents[12], and information visualization[26].

Automated collaborative filtering systems work by collecting human judgments (known as ratings) for items in a given domain and matching together people who share the same information needs or the same tastes. Users of a collaborative filtering system share their analytical judgments and opinions regarding each item that they consume so that other users of the system can better decide which items to consume. In return, the collaborative filtering system provides useful personalized recommendations for interesting items.

Collaborative filtering provides three key additional advantages to information filtering that are not provided by content-based filtering: (i) support for filtering items whose content is not easily analyzed by automated processes; (ii) the ability to filter items based on quality and taste; and (iii) the ability to provide serendipitous recommendations.

First of all, in collaborative filtering, humans determine the relevance, quality, and interest of an items in the information stream. As a result, filtering can be performed on items that are hard to analyze with computers, such as movies, ideas, feelings, people, and politicians.

Second, collaborative filtering systems can enhance information filtering systems by measuring, in dimensions beyond that of simple content, how well an item meets a user's need or interests. Humans are capable of analyzing on dimensions such as quality or taste, which are very hard for computer processes. A content-based search of the Associated Press could retrieve all articles related to Minnesota Governor Jesse Ventura, but by combining content filtering with collaborative filtering, a search could return only those relevant articles that are well-written!

Finally, a collaborative filtering system will sometimes make serendipitous recommendations—recommending items that are valuable to the user, but do not contain content that the user was expecting. We have found that serendipitous recommendations occur frequently in the movie domain, with the collaborative filtering system accurately recommending movies that a user would never have considered otherwise.

The potential for collaborative filtering to enhance information filtering tools is great. However, to reach the full potential, it must be combined with existing content-based information filtering technology. Collaborative filtering by itself performs well predicting items that meet a user's interests or tastes, but is not well-suited to locating information for a specific content information need.

In this paper, we present an algorithmic framework for performing collaborative filtering and examine empirically how existing algorithm variants perform under this framework. We present new, effective enhancements to existing prediction algorithms and finally conclude with a set of recommendations for selection of prediction algorithm variants.

2 Problem Space

The problem of automated collaborative filtering is to predict how well a user will like an item that he has not rated

	Star Wars	Hoop Dreams	Contact	Titanic
Joe	5	2	5	4
John	2	5		3
Al	2	2	4	2
Nathan	5	1	5	?

Figure 1: Collaborative filtering can be represented as the problem of predicting missing values in a user-item matrix. This is an example of a user-item rating matrix where each filled cell represents a user's rating for an item. The prediction engine is attempting to provide Nathan a prediction for the movie 'Titanic'.

given a set of historical preference judgments for a community of users.

Preference judgments can be explicit statements recorded from the user or implicit measures that are inferred from available data on user activity. Explicit measures are generally a single numeric summary rating for each item[17, 21, 9], with high values representing a strong interest in an item and low values representing a strong disinterest. Users are generally instructed to rate an item as they would like to have seen that item predicted.

Implicit ratings are commonly derived from data sources such as purchase records or web logs, thereby leveraging data already collected for other purposes. Other sources of implicit preference data that have been explored are time spent reading[15] and URL references in Usenet postings[23]. Little work has been done on automatic collaborative filtering using implicit ratings although one technique that has been considered for supporting implicit ratings is to map implicit measures such as time-spent-reading into an explicit rating scale, and then use previously proven collaborative filtering algorithms. For the purposes of this article, we will consider explicit user ratings on a scale of 1 to 5.

A *prediction engine* collects all the ratings and uses collaborative filtering technology to provide predictions. An active user provides the prediction engine with a list of items, and the prediction engine returns a list of predicted ratings for those items. Most prediction engines[17, 21, 9] also provide a *recommendation* mode, where the prediction engine returns the top n highest predicted items for the active user from the database. In order to provide a fluid user interface, a prediction engine has performance constraints. Latency of a prediction request must be less than 1 second and prediction engines must often support throughput of several hundred prediction requests per second[25].

The problem space can be formulated as a matrix of users versus items, with each cell representing a user's rating on a specific item. Under this formulation, the problem is to predict the values for specific empty cells. In collaborative filtering, this matrix is generally very sparse, since each user will only have rated a small percentage of the total number of items. Figure 1 shows a simplified example of a user-rating matrix where predictions are being computed for movies.

The most prevalent algorithms used in collaborative filtering are what we call the *neighborhood-based methods*. In neighborhood-based methods, a subset of appropriate users are chosen based on their similarity to the active user, and a weighted aggregate of their ratings is used to generate predictions for the active user. Other algorithmic methods that have been used are Bayesian networks[5], singular value decomposition with neural net classification[4], and induction rule learning[3]. As an example of a neighborhood based method, consider Figure 1 again. We wish to predict how Nathan will like the movie "Titanic." Joe is Nathan's best neighbor, since the two of them have agreed closely on all movies that they have both seen. As a result, Joe's opinion of the movie Titanic will influence Nathan's prediction the most. John and Al are not as good neighbors because both of them have disagreed with Nathan on certain movies. As a result, they will influence Nathan's predictions less than Joe.

In this paper we will explore the space of neighborhood-based collaborative filtering methods and describe some new better performing algorithms that we have developed.

Neighborhood-based methods can be separated into three steps.

1. Weight all users with respect to similarity with the active user.
2. Select a subset of users to use as a set of predictors (possibly for a specific item).
3. Normalize ratings and compute a prediction from a weighted combination of selected neighbors' ratings.

Within specific systems, these steps may overlap or the order may be slightly different.

We will begin by discussing related work in the field of collaborative filtering and, when applicable, examining which techniques were used to implement the three steps of neighborhood-based methods.

3 Related Work

The concept of collaborative filtering is relatively new, and descends from work in the area of information filtering[1]. The term *collaborative filtering* was coined by Goldberg et al.[8], who were the first to publish an account of using collaborative filtering techniques in the filtering of information. They built a system for filtering email called Tapestry which allowed users to annotate messages. Annotations became accessible as virtual fields of the messages, and users could construct filtering queries which accessed those fields. Users could then create queries such as "show me all office memos that Bill thought were important." The collaborative filtering provided by Tapestry was not automated, and required users to construct complex queries in a special query language designed for the task.

GroupLens[17, 11] first introduced an automated collaborative filtering system using a neighborhood-based algorithm. GroupLens provided personalized predictions for Usenet news articles. The original GroupLens system used Pearson correlations to weight user similarity, used all available correlated neighbors, and computed a final prediction by performing a weighted average of deviations from the neighbor's mean:

$$p_{a,i} = \overline{r}_a + \frac{\sum_{u=1}^{n} (r_{u,i} - \overline{r}_u) * w_{a,u}}{\sum_{u=1}^{n} w_{a,u}} \quad (1)$$

$p_{a,i}$ represents the prediction for the active user a for item i. n is the number of neighbors and $w_{a,u}$ is the similarity

weight between the active user and neighbor u as defined by the Pearson correlation coefficient:

$$w_{a,u} = \frac{\sum_{i=1}^{m}(r_{a,i} - \overline{r}_a) * (r_{u,i} - \overline{r}_u)}{\sigma_a * \sigma_u} \quad (2)$$

The Ringo music recommender[21] and the Bellcore Video Recommender[9] expanded upon the original GroupLens algorithm. Ringo claimed better performance by computing similarity weights using a *constrained* Pearson correlation coefficient: $w_{a,u} = \frac{\sum_{i=1}^{m}(r_{a,i}-4)*(r_{u,i}-4)}{\sigma_a * \sigma_u}$ where 4 was chosen because it was the midpoint of their seven-point rating scale. Ringo limited membership in a neighborhood by only selecting those neighbors whose correlation was greater than a fixed threshold, with higher thresholds resulting in greater accuracy, but reducing the number of items for which Ringo was able to generate predictions for. To generate predictions, Ringo computed a weighted average of ratings from all users in the neighborhood.

The Bellcore Video Recommender[9] used Pearson correlation to weight a random sample of neighbors, selected the best neighbors, and performed a full multiple regression on them to create a prediction.

Breese et al.[5] performed an empirical analysis of several variants of neighborhood-based collaborative filtering algorithms. For similarity weighting, Pearson correlation and cosine vector similarity were used, with correlation being found to perform better.

Other collaborative filtering systems have been developed which do not make use of neighborhood-based prediction algorithms. The Fab system[2] integrates content and collaborative filtering by identifying user tastes via content profiles, collecting ratings from users, and forwarding highly rated documents to users with similar profiles. Other systems include the PHOAKS system for recommending web resources[23], Referral Web[10], and active collaborative filtering by Maltz and Ehrlich[13].

4 Methodology

4.1 Data & Experimental Technique

In order to compare the results of different neighborhood based prediction algorithms, we ran a prediction engine using historical ratings data collected for purposes of anonymous review from the MovieLens movie recommendation site[6]. The historical data consisted of 122,176 ratings from 1173 users, with every user having at least 20 ratings. 10% of the users were randomly selected to be the test users. From each user in the test set, ratings for 5 items were withheld, and predictions were computed for those 5 items using each variant of the tested neighborhood based prediction algorithms.

For each item predicted, the highest ranking neighbors that have rated the item in question are used to compute a prediction (they form the user's *neighborhood* for that item). Note that this means that a user may have a different neighborhood for each item. All users in the database are examined as potential neighbors for a user—no sampling is performed.

The quality of a given prediction algorithm can be measured by comparing the predicted values for the withheld ratings to the actual ratings.

4.2 Metrics

There are three key dimensions on which the quality of a prediction algorithm can be measured.

Coverage. Coverage is a measure of the percentage of items for which a recommendation system can provide predictions. A basic coverage metric is the percentage of items for which predictions are available. Common system features that can reduce coverage are small neighborhood sizes and sampling of users to find neighbors. We compute coverage as the percentage of items over all users for which a prediction was requested and the system was able to produce a prediction. Unless otherwise noted, all experimental results demonstrated in this paper had maximal coverage. Maximal coverage may be slightly less than perfect (99.8% in our experiments) because there may be no ratings in the data for certain items, or because very few people rated an item, and those that did had zero correlations with the active user.

Accuracy. Many metrics have been proposed for assessing the accuracy of a collaborative filtering system. They divide into two main categories: statistical accuracy metrics and decision-support accuracy metrics. 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. Mean absolute error (MAE) has been used previously to measure prediction engine performance by Shardanand & Maes[21] and Sarwar, et al[20]. Other metrics that have been used are root mean squared error[20] and correlation between ratings and predictions[9, 11, 20]. All of the above metrics were computed on results data and generally provided the same conclusions, so we only report mean absolute error.

Decision-support accuracy metrics evaluate how effectively predictions help a user select high-quality items from the item set. They are based on the observation that, for many users, filtering is a binary process. The user either will or will not view the movie. If this is the case, then the difference between a prediction of 1.5 and 2.5 is irrelevant if the user only views movies recommended with a prediction of 4 or more. For our decision support accuracy measure, we use ROC sensitivity.

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. 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 prediction score threshold above which the movie is accepted. For example, a particular point might correspond to setting the filter at a prediction of exactly 4—watch any movie recommended with a prediction of 4 or above. 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. We consider the "goodness" model one where ratings of 4 and 5 indicate signal and 1, 2, and 3 are noise, which we call

Component	Variants Tested
Similarity Weight	Pearson Correlation
	Spearman Correlation
	*Vector Similarity
	Entropy
	Mean-squared-difference
Significance Weighting	No significance weighting
	n/50 weighting
Variance Weighting	None
	(variance – $variance_{min}$) / ($variance_{max}$ – $variance_{min}$)
Selecting Neighborhoods	Weight Thresholding
	Best-n neighbors
	Combined
Rating Normalization	No normalization
	Deviation from mean
	Z-score

Figure 2: List of prediction algorithm components tested and the variants of each component that were tested. *Vector Similarity was considered, but was rejected due to previous work[5].

ROC-4. The range of ROC sensitivity is 0 to 1, where 0.5 is random and 1 is perfect. The ROC curve is related to the recall-fallout curve—for a more in-depth explanation, please see [24, 22]

4.3 Experimental Design

This paper presents conclusions derived from empirical analysis of prediction algorithm components. The components tested are listed in Figure 2 along with the variations of each component that were tested. For each component, the performance using each of the variations was measured. All components except the one being measured were held constant to ensure that the results reflected the differences in the component being tested. In each case, the variations of a component were tested on the best performing algorithm at the time of the experiment.

5 Weighting Possible Neighbors

5.1 Similarity Weighting

The first step in neighborhood-based prediction algorithms is to weight all users with respect to similarity with the active user. When you are given recommendations for movies or books, you are more likely to trust those that come from people who have historically proven themselves as providers

	MAE		ROC-4	
	Spearman	Pearson	Spearman	Pearson
Top-50 neighbors	**.7678**	.7687	.7288	.7288
Best non-personalized average	.7982		.7127	

Figure 3: Comparative performance of two different similarity metrics, Pearson correlation and Spearman correlation. The best-50 neighbors are selected using each algorithm and a prediction is computed from those neighbors. Spearman correlation performs as well as Pearson correlation, but is not subject to model assumptions.

of accurate recommendations. Likewise, when automatically generating a prediction, we want to weight neighbors based on how likely they are to provide an accurate prediction.

Several different similarity weighting measures have been used. The most common weighting measure used is the *Pearson correlation coefficient*. Pearson correlation measures the degree to which a linear relationship exists between two variables.

The Pearson correlation coefficient (Equation 2) is derived from a linear regression model that relies on a set of assumptions regarding the data, namely that the relationship must be linear, and the errors must be independent and have a probability distribution with mean 0 and constant variance for every setting of the independent variable[14]. When these model assumptions are not satisfied, Pearson correlation becomes a much less accurate indicator of similarity. It is not uncommon for these model assumptions to be violated in collaborative filtering data.

Spearman rank correlation coefficient (Equation 3) is similar to Pearson, but does not rely on model assumptions, computing a measure of correlation between ranks instead of rating values.

$$w_{a,u} = \frac{\sum_{i=1}^{m}(rank_{a,i} - \overline{rank}_a) * (rank_{u,i} - \overline{rank}_u)}{\sigma_a * \sigma_u} \quad (3)$$

In our experiments, we have found that Spearman correlation performs similarly to Pearson correlation. Figure 3 shows results for both Spearman and Pearson correlation, using several different neighborhood selection mechanisms that are discussed in the section *Selecting Neighborhoods*. Note that the results are very close between Spearman and Pearson. The large number of tied rankings (there are only five distinct ranks for each user) results in a degradation of the accuracy of Spearman correlations. Future work could determine if increasing the size of the ranking scale increases the difference in accuracy between Spearman and Pearson.

Other similarity measures include the vector similarity "cosine" measure, the entropy-based uncertainty measure, and the mean-squared difference algorithm. The vector similarity measure has been shown to be successful in information retrieval[19], however Breese has found that vector similarity does not perform as well as Pearson correlation in collaborative filtering[5]. The measure of association based on entropy[16] uses conditional probability techniques to measure the reduction in entropy of the active user's ratings that results from knowing the another user's ratings. In our tests, entropy has not shown itself to perform as well as Pearson correlation. We also found that the mean-squared difference algorithm, introduced in the Ringo system[21], also did not perform well compared to Pearson correlation.

	MAE		ROC-4	
	N/50 devaluing	None	N/50 devaluing	None
Top-50 neighbors	**.7678 (p = 0.02)**	.7906	**.7288**	.7211

Figure 4: Small numbers of co-rated items leads to many misleadingly high correlations. To account for this, correlations based on less than 50 co-rated items are multiplied by a significance weight of $n/50$ where n is the number of commonly rated items. The result is a significantly large increase in accuracy.

5.2 Significance Weighting

One of the issues that has not been addressed in previously published systems is the amount of trust to be placed in a correlation with a neighbor. In our experience with collaborative filtering systems, we have found that it was common for the active user to have highly correlated neighbors that were based on a very small number of co-rated items. These neighbors that were based on tiny samples (often three to five co-rated items) frequently proved themselves to be terrible predictors for the active user. The more data points that we have to compare the opinions of two users, then the more we can trust that the computed correlation is representative of the true correlation between the two users. We hypothesized that the accuracy of prediction algorithms would be improved if we were to add a correlation significance weighting factor that would devalue similarity weights that were based on a small number of co-rated items. For our experiments, if two users had fewer than 50 commonly rated items, we applied a significance weight of $n/50$, where n is the number of co-rated items. If there were more than 50 co-rated items, then a significance weight of 1 was applied. In this manner, correlations with small numbers of co-rated items are appropriately devalued, but correlations with 50 or more commonly co-rated items are not dependent on the number of co-rated items. Figure 4 compares the results of a Spearman correlation-based prediction algorithm with and without the devaluing term. As can be seen from the figure, applying the significance weighting increased the accuracy of the prediction algorithm by a relatively large amount. Similar results were found for Pearson correlation.

5.3 Variance Weighting

All the similarity measures described above treat each item evenly in a user to user correlation. However, knowing a user's rating on certain items is more valuable than others in discerning a user's interest. For example, we have found that the majority of MovieLens users have rated the movie "Titanic" highly. Therefore knowing that two users rated Titanic high tells us very little about the shared interests of those two users. Opinions on other movies have been known to distinguish users' tastes. The movie "Sleepless in Seattle" has shown itself to separate those users who like action movies from those who like romance movies. Knowing that two people agree on "Sleepless in Seattle" tells us a lot more about their shared interests than Titanic would have. We hypothesized that giving the distinguishing movies more influence in determining a correlation would improve the accuracy of the prediction algorithm. To achieve this, we modified the Pearson correlation algorithm to incorporate an item-variance weight factor. The Pearson correlation(Equation 2) can be represented as the covariance of two users' ratings, after the ratings have been scaled to z-scores (mean 0, standard deviation 1). This is shown in Equation 4.

	MAE	ROC-4
Pearson – no variance weighting	0.7740	0.7287
Pearson – variance weighting	0.7774	0.7278

Figure 5: Performance of the prediction algorithm with and without variance weighting. Contrary to our hypothesis, variance weighting did not improve the accuracy of the predictions.

$$w_{a,u} = \frac{\sum_{i=1}^{m} z_{a,i} * z_{u,i}}{m} \quad (4)$$

By incorporating a variance weight term, we will increase the influence of items with high variance in ratings and decrease the influence of items with low variance. The new correlation, incorporating the variance weight term is shown in equation 5.

$$w_{a,u} = \frac{\sum_{i=1}^{m} v_i * z_{a,i} * z_{u,i}}{\sum_{i=1}^{m} v_i} \quad (5)$$

We computed an item variance weight as $v_i = \frac{var_i - var_{min}}{var_{max}}$ where $var_i = \frac{\sum_{u=1}^{n} (r_{u,i} - \overline{r_i})^2}{n-1}$, and var_{min} and var_{max} respectively are the minimum and maximum variances over all items. Contrary to our initial hypothesis, applying variance weighting terms to correlation had no significant effect on the accuracy of the prediction algorithm. These results are shown in Figure 5. We are currently examining different ways of accounting for item variance to determine if the hypothesis is wrong or our implementation is flawed. One explanation is that our variance weighting scheme does not take into account the fact that a user who disagrees with the popular feeling provides a lot of information.

6 Selecting Neighborhoods

After you have assigned similarity weights to users in the database, you must determine which other users' data will be used in the computation of a prediction for the active user. It is useful, both for accuracy and performance, to select a subset of users (the *neighborhood*) to use in computing a prediction instead of the entire user database[21]. This can be seen in Figure 6 which shows how the mean absolute error of the prediction increases as the neighborhood size is increased. In addition, commercial collaborative filtering systems are beginning to handle millions of users, making consideration of every neighbor infeasible. The system must select the best neighbors, discarding the remaining users.

Another consideration in selecting neighborhoods suggested by Breese[5] is that high correlates (such as those with correlations greater than 0.5) can be exceptionally more valuable as predictors than those with lower correlations.

Two techniques, correlation-thresholding and best-n-neighbors, have been used to determine how many neighbors to select. With both of these techniques, contributions from the few exceptionally valuable high correlations tend to get lost in the noise from the many lower correlations when the algorithms are configured to give acceptable coverage.

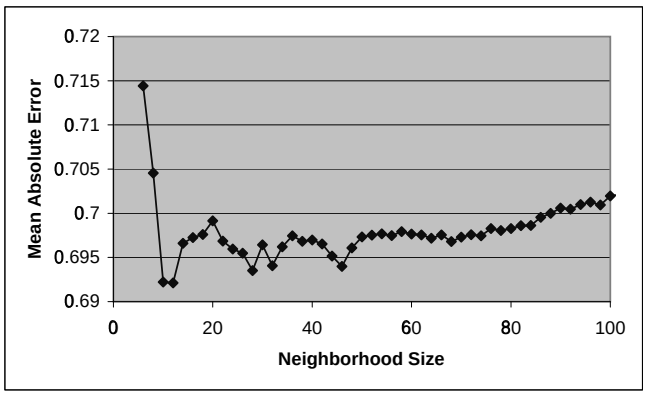

Figure 6: As you increase the size of the neighborhood, the quality of the prediction decreases, indicated by mean absolute error. This experiment was performed using Pearson correlation with $n/50$ significance weight.

Algorithm	coverage	MAE	ROC-4
All nbors	99.8%	0.776069	0.72593
min-abs-corr=0.1	99.4%	0.770466	0.73011
min-abs-corr=0.2	84.0%	0.792018	0.73942
min-abs-corr=0.3	60.9%	0.757773	0.76919
min-abs-corr=0.4	42.9%	0.781000	0.76581
min-abs-corr=0.5	19.6%	0.762424	0.82114

Figure 7: Increasing the absolute correlation weight thresholds to limit neighborhood sizes quickly results in loss of prediction coverage.

The first technique, used by Shardanand and Maes[21], is to set an absolute correlation threshold, where all neighbors with absolute correlations greater than a given threshold are selected. Setting a high threshold limits your neighborhood to containing very good correlates, but for many users high correlates are not available, resulting in a small neighborhood that cannot provide prediction coverage for many items. This is demonstrated in Figure 7, which demonstrates the loss of prediction coverage when using an absolute correlation threshold. Setting a lower correlation threshold results in a large number of lower correlates, nullifying the purpose of thresholding. This can also be seen in Figure 7, where an absolute weight threshold of 0.1 provides approximately the same accuracy as using all the available users in the dataset.

The second technique is to pick the best n correlates for a given n. This technique performs reasonably well, as it does not limit prediction coverage. However, picking a larger n will result in too much noise for those who have high correlates. Picking a smaller n can cause poor predictions for those users who do not have any high correlates, although in our experiments, this effect did not occur until the neighborhood size was reduced below 10. Both of these effects can been seen in Figure 6.

Figure 8 shows an overview of the best performing neighborhood selection algorithms. The weight thresholding algorithm using a high threshold performed the best, but sacrificed too much coverage. The best-n method provided the best performance with no loss in coverage. Combining a low similarity weight threshold with best-n techniques did not result in a significant improvement over using best-n. We believe that this is because there is little difference in predictive value between extremely low correlates (less than 0.1) and moderately low correlates (0.1–0.3).

Algorithm	Coverage	MAE	ROC-4
min-abs-corr=0.3	60.9%	0.757773	0.76919
min-abs-corr=0.5	19.6%	0.762424	0.82114
min-abs-corr=0.1, nnbors=20	99.4%	0.763836	0.73447
min-abs-corr=0.1, nnbors=40	99.4%	0.765492	0.73134
max-nbors=20	99.8%	0.765686	0.73228

Figure 8: The best performing neighborhood selection algorithms according to mean absolute error. Using a similarity weight threshold of 0.3 resulted in the lowest amount of error, but had an unacceptable loss of 39% coverage.

7 Producing a Prediction

Once the neighborhood has been selected, the ratings from those neighbors are combined to compute a prediction, after possibly scaling the ratings to a common distribution. The basic way to combine all the neighbors' ratings into a prediction, as used in Ringo[21], is to compute a weighted average of the ratings, using the correlations as the weights. The basic weighted average makes an assumption that all users rate on approximately the same distribution.

The approach taken by GroupLens[17] was to compute the average deviation of a neighbor's rating from that neighbor's mean rating, where the mean rating is taken over all items that the neighbor has rated. The *deviation-from-mean* approach is demonstrated in Equation 1. The justification for this approach is that users may rating distributions centered around different points. One user may tend to rate items higher, with good items getting 5s and poor items getting 3s, while other users may give primarily 1s, 2s, and 3s. Intuitively, if a user infrequently gives ratings of 5, then that user should not receive many predictions of 5 unless they are extremely significant. The average deviation from the mean computed across all neighbors is converted into the active user's rating distribution by adding it to the active user's mean rating.

An extension to the GroupLens algorithm is to account for the differences in spread between users' rating distributions by converting ratings to z-scores, and computing a weighted average of the z-scores (Equation 6).

$$p_{a,i} = \overline{r}_a + \sigma_a * \frac{\sum_{u=1}^{n} \frac{r_{u,i} - \overline{r}_u}{\sigma_u} * w_{a,u}}{\sum_{u=1}^{n} w_{a,u}} \qquad (6)$$

Figure 9 compares the performance of the three rating normalization techniques. The bias-from-mean approach performs significantly better than the non-normalized rating approach (p = 0), however, z-scores do not perform significantly better than the bias-from-mean approach, suggesting that differences in spread between users' rating distributions do not effect predictions.

Normalization Algorithm	MAE	ROC-4
Average z-score	0.760290	0.73221
Average deviation from mean	0.765686	0.73228
Simple weighted average	1.162838	0.63840

Figure 9: Performance of prediction algorithm with and without normalization. Note the significant increase in accuracy from using deviation-from-mean normalization.

Algorithm	MAE	ROC-4
average	0.8277	0.6786
bias-from-mean average	0.7982	0.7127
z_score_average	0.8012	0.7049

Figure 10: Deviation-from-mean average performs much better than the average rating as a non-personalized prediction algorithm.

8 Conclusions

Collaborative filtering is an exciting new approach to filtering information that can select and reject items from an information stream based on qualities beyond content, such as quality and taste. It has the potential to enhance existing information filtering and retrieval techniques.

In this paper we have presented an algorithmic framework that breaks the collaborative prediction process into components, and we provide empirical results regarding variants of each component, as well as present new algorithms that enhance the accuracy of predictions.

The empirical conclusions in this paper are drawn from an analysis of historical data collected from an operational movie prediction site. The data is representative of a large set of rating-based systems, where the domain of predictions is high volume targeted entertainment with a generally high level of quality control. Domains of this criteria include movies, videos, books, & music. There is reason to believe that these results are generalizable to other prediction domains, but we do not yet have empirical results to prove it. Our algorithmic recommendations are certainly a good place to start when exploring a new and different prediction domain.

We have made new contributions in each of the three steps of the neighborhood-based prediction algorithm. We showed that Spearman correlation performed as well as Pearson correlation and because it is not dependent on model assumptions, it should perform consistently across diverse datasets. We demonstrated that incorporating significance weighting by devaluing correlations that are based on small numbers of co-rated items provided a significant gain in prediction accuracy. While we hypothesized that decreasing the contributions of items which had a low rating variance across all users would increase predictions accuracy, it proved false, with variance weights decreasing the prediction accuracy. Best-n neighbors proved to be the best approach to selecting neighbors to form a neighborhood. Finally, deviation-from-mean averaging was shown to increase prediction accuracy significantly over a normal weighted average, while z-score averaging provided no significant improvements.

For those who are considering using a neighborhood-based prediction algorithm to perform automated collaborative filtering, we have the following recommendations: If your rating scale consists of a small number of discrete ranks (i.e. integers 1–5, 1–7, or 1–20), use Spearman correlation as your similarity weighting measure. If your rating scale is not discrete, but continuous, you may want to consider Pearson correlation. If your rating scale is binary or unary, you will have to consider a different approach — see Breese[5] for more information. It is important to use a significance weight to devalue correlates with small numbers of co-rated items as it will often give you a larger gain in accuracy than your choice of similarity algorithm. Finally, users will rate on slightly different scales, so use the deviation-from-mean approach to normalization.

In the progress of examining personalized algorithms, we also discovered a much more accurate non-personalized average algorithm. Automated collaborative filtering systems use non-personalized average algorithms to provide predictions when not enough is known about the user to provide a personalized prediction. The normal approach is to compute the average rating of the item being predicted over all users (Equation 7).

$$p_{a,i} = \frac{\sum_{u=1}^{n} r_{u,i}}{n} \tag{7}$$

However, we have found that computing a deviation-from-mean average over all users (Equation 8)

$$p_{a,i} = \overline{r}_a + \frac{\sum_{u=1}^{n} (r_{u,i} - \overline{r}_u)}{n} \tag{8}$$

results in a much more accurate non-personalized prediction as is demonstrated in Figure 10.

9 Future Work

This paper presents results taken from an empirical analysis of one class of prediction domain. Ideally, the next step would be to apply the framework to a diverse set of prediction domains, and determine what results hold generally across prediction domains. Unfortunately, sufficiently large test sets containing a rating range of more than two are not readily available.

Further success of collaborative filtering systems require that collaborative filtering systems integrate with existing content-based filtering and retrieval technology. This could happen at the algorithmic level, integrating content weighting schemes such as text vectors[19] with collaborative filtering weights, or as a pipeline process, such as performing first selecting a set of documents based on content and ranking them based on collaborative filtering. Some initial work has been done in this field[2], but much more work remains to be done.

Another challenge is scaling prediction algorithms to handle extremely large datasets. It may become impractical to compute correlations between the active user and all other users in the data, except in an infrequent off-line manner. Techniques of dimensionality reduction are currently being investigated in an attempt to reduce the amount of online computation. Example techniques are sampling of users, singular value decomposition (used in Latent Semantic Analysis[7]), and stereotypical modeling[18].

Acknowledgments Funding for this research was provided in part 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

[1] ACM. Special issue on information filtering. *Communciations of the ACM*, 35(12), December 1992.

[2] Marko Balabanovíc and Yoav Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, March 1997.

[3] Chumki Basu, Haym Hirsh, and William Cohen. Recommendation as classification: using social and content-based information in recommendation. In *Proceedings of the 1998 Workshop on Recommender Systems*, pages 11–15. AAAI Press, August 1998.

[4] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proceedings of the 1998 Workshop on Recommender Systems*. AAAI Press, August 1998.

[5] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 43–52, San Francisco, July 24–26 1998.

[6] Brent J. Dahlen, Joseph A. Konstan, Jon Herlocker, Nathaniel Good, Al Borchers, and John Riedl. Jump-starting movielens: User benefits of starting a collaborative filtering system with "dead data". Technical Report TR 98-017, University of Minnesota, 1998.

[7] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[8] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, December 1992.

[9] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, pages 194–201, 1995.

[10] Henry Kautz, Bart Selman, and Mehul Shah. Referral Web: Combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, March 1997.

[11] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3):77–87, March 1997.

[12] Pattie Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):30–40, July 1994.

[13] David Maltz and Kate Ehrlich. Pointing the way: Active collaborative filtering. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, pages 202–209, 1995.

[14] James T. McClave and Frank H. Dietrich II. *Statistics*. Dellen Publishing Company, 1988.

[15] Masahiro Morita and Yoichi Shinoda. Information filtering based on user behavior analysis and best match text retrieval. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 272–281, 1994.

[16] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, New York, NY, 1986.

[17] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM CSCW'94 Conference on Computer-Supported Cooperative Work*, pages 175–186, 1994.

[18] Elaine Rich. User modeling via stereotypes. *Cognitive Science*, 3:335–366, 1979.

[19] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retreival. *Information Processing & Management*, 24(5):513–523, 1988.

[20] Badrul M. Sarwar, Joseph A. Konstan, Al Borchers, Jon Herlocker, Brad Miller, and John Riedl. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In *Proceedings of 1998 Conference on Computer Supported Collaborative Work*, November 1998.

[21] Upendra Shardanand and Patti Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, pages 210–217, 1995.

[22] John A. Swets. Measuring the accuracy of diagnostic systems. *Science*, 240(4857):1285–1289, June 1988.

[23] Loren Terveen, Will Hill, Brian Amento, David McDonald, and Josh Creter. PHOAKS: A system for sharing recommendations. *Communications of the ACM*, 40(3):59–62, March 1997.

[24] C. J. van Rijsbergen. *Information Retrieval*, chapter 7. Butterworths, 1979.

[25] NetPerceptions Inc web site. http://www.netperceptions.com/.

[26] James A. Wise, James J. Thomas, Kelly Pennock, David Lantrip, Marc Pottier, Anne Schur, and Vern Crow. Visualizing the non-visual: Spatial analysis and interaction with information from text documents. In *IEEE Information Visualization '95*, pages 51–58. IEEE Computer Soc. Press, 30–31 October 1995.