Collaborative Filtering with Decoupled Models for Preferences and Ratings

Rong Jin[1], Luo Si[1], ChengXiang Zhai[2] and Jamie Callan[1]

[1] School of Computer Science, Carnegie Mellon University

[2] Department of Computer Science, University of Illinois at Urbana-Champaign

ABSTRACT

In this paper, we describe a new model for collaborative filtering. The motivation of this work comes from the fact that two users with very similar preferences on items may have very different rating schemes. For example, one user may tend to assign a higher rating to all items than another user. Unlike previous models of collaborative filtering, which determine the similarity between two users only based on their rating performance, our model treats the user's preferences on items separately from the user's rating scheme. More specifically, for each user, we build two separate models: a preference model capturing which items are favored by the user and a rating model capturing how the user would rate an item given the preference information. The similarity of two users is computed based on the underlying preference model, instead of the surface ratings. We compare the new model with several representative previous approaches on two data sets. Experiment results show that the new model outperforms all the previous approaches that are tested consistently on both data sets.

Categories & Subject Descriptors:

H.3.3 [**Information Search and Retrieval]:** Information Search and retrieval—*Information Filtering*

General Terms

Application, Algorithms.

Keywords

Collaborative filtering, probabilistic model, preference model, rating model

1. INTRODUCTION

With the rapid growth of digital data repositories and the overwhelming supply of on-line information on the Internet, it becomes more and more important to separate important information from those nuisance data. Information filtering refers to the task where useful information is separated from the irrelevant information. In this paper, we focus on the task of collaborative filtering, which is to predict the utility of items for a particular user based on the ratings of information items given by many other users. The fact that collaborative filtering does not rely on any content information about the items or descriptions of users, but only depends on the preference patterns of users makes it more general than other tasks such as ad hoc information retrieval and content-based filtering [6,7]. In many cases, collaborative filtering reflects a more realistic setup, particularly in the web environment, where descriptions of items and users are not available due to the privacy issue.

Many algorithms have been proposed to deal with the collaborative filtering problem [1, 2, 3, 4, 5, 8, 12]. According to [1], most collaborative filtering algorithms can be categorized into two classes: Memory-based algorithms and model-based algorithms. The memory-based algorithms first find the users from the training database that are most similar to the current test user in terms of the rating pattern, and then combine the ratings given by those similar users to obtain the prediction for the test user. The major approaches within this category are the Pearson-correlation based approach [4], the vector similarity based approach [1], and the extended generalized vector space model [3]. Model-based approaches group together different users in the training database into a small number of classes based on their rating patterns. In order to predict the rating of a test user on a particular item, we can simply categorize the test user into one of the predefined user classes and use the predicted class as the prediction for the test user. Proposed algorithms within this category include a Bayesian network approach [1], and the aspect model [5]. Compared with the memory-based approaches, the model-based approaches only have to store the profiles of models and therefore are much more efficient than storing the whole user database. On the other hand, the offline computation of memory-based approaches can be much cheaper than the model-based approaches, which often require the use of the Expectation-Maximization (EM) [11] algorithm, variation method and other sophisticated methods to combat the complexity of computation. Furthermore, model-based approaches tend to assume that a small number of user classes is sufficient for modeling the rating patterns of users, thus imply a loss of diversity among users, which may be very important in helping predict the ratings of a test user. Indeed, according to the previous studies on the comparison of memory-based approaches and model-based approaches, memory-based approaches such as the correlation method performs very close to those complicated model-based methods such as a Bayesian network approach and an aspect model [1]. Because both memory-based and model-based models

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM '03, November 3--8, 2003, New Orleans, Louisiana, USA

Copyright 2003 ACM 1-58113-723-0//03/0011…$5.00.

have their own advantages and disadvantages, there are hybrid approaches that try to unify these two types of approaches into a single model. The approach 'Personal Diagnosis' [12] belongs to this category, which appears to outperform previous model-based and memory-based approaches.

There are two major issues involved in the collaborative filtering task. The first one is the missing data issue. Usually most users would rate only a small number of items within the whole item set. Therefore the votes in the intersection of items rated by both users will be small, which may lead to a poor estimate of the similarities between users. Special treatments are needed to deal with the missing ratings. For example, in the correlation approach, default ratings are introduced for those unrated items [1]. For some probabilistic models, an extra rating category named 'no-rating' is introduced in the case when ratings are missing [1]. The second issue arose out of the observation that users with similar preferences on items may still rate items differently. Indeed, in the previous study, the correlation approach usually outperforms the vector similarity approach significantly [1]. One major difference between these two approaches is that, for the correlation approach, the similarity (i.e. the Pearson Correlation Coefficients) between two users are measured based on the relative ratings, namely the original ratings subtracted by the average rating of each user, while for vector similarity approach, the original ratings are used for measuring the user similarity. In the comparison of these two approaches in [1], the correlation method outperforms the vector similarity method significantly, particularly in terms of the MSE metric. This fact indicates that, due to the variance in the rating behavior of different users, the rating information may not be able to indicate the preference information directly. Therefore, an extra transformation is needed to convert the rating information into preference information. In the correlation method, the transformation is accomplished by subtracting the original ratings from the averaged ratings of each user. However, this simple transformation does not take into account the whole rating distribution.

In this paper, we propose a new probabilistic model that *directly* addresses the issue that users with similar preference pattern(s) may adopt different rating patterns. We exploit model smoothing to deal with the problem of sparse data and missing values. Although some previous probabilistic models (e.g., the two-side clustering approach [5]) have indirectly captured the difference between user ratings and user preferences, our approach is more direct –we *explicitly* convert ratings into preferences.

The core part of our algorithm is a probabilistic mechanism that is able to transform the rating information of items into the likelihood for items to be preferred by a user. With this mechanism, instead of computing the similarity between users based on the observed ratings, we can compare users based on their underlying preference patterns on the items. Furthermore, with the same conversion mechanism, we can also predict the rating for a new user on an item by first computing the underlying preference of the user on the item and then computing the most likely rating category for that item. This new framework is similar to the model-based approaches in that global models will be computed and computation is efficient.

The rest of this paper is arranged as follows: Section 2 discusses the details of this new framework for collaborative filtering. The empirical study is presented in Section 3. Section 4 concludes this work and discusses the potential future work.

2. A New Framework for Collaborative Filtering

A major problem with the traditional memory-based approach is that, a user's rating behavior and preference behavior are tangled with each other. As a result, the user similarity is often computed based on the surface rating patterns, and a direct combination of other users' surface ratings is used to predict the rating for a new user. Since users may have different rating tendencies and two users with different preferences may accidentally have similar surface rating patterns, it is desirable to separate the preference information from the rating information, so that we can explicitly find similar users in terms of either their preference patterns or the rating patterns. In this paper, we propose a memory-based probabilistic model, which decouples the rating pattern and the preference pattern of a user. We treat each user within the training database as a separate model (or expert), and a database of users serve as an ensemble of models. A Bayesian approach can then be used to predict the rating for a new user by combining the predictions given by all the expert models.

In this model, in order to predict the ratings for test users, four steps are required:

1) First, a "conversion model" is computed for all users, including both the training and test users. Such a conversion model makes it possible to estimate the likelihood of user preferring an object based on the observed rating pattern of the user. This is a crucial step in this new model, because, with this conversion, we can explicitly estimate the preference similarity of users without worrying about the variation of the rating behavior among different users.
2) Second, the preference similarities between the test user and the training users are computed using the estimated preference information. As in the work by Pennock et al. [12], the similarity between two users is computed based on the likelihood of mistaking one user as the other.
3) Third, the preference patterns on the un-rated objects for the test user are computed as the combination of preference patterns of training users weighted by their preference similarities. This is essentially Bayesian model averaging.
4) Finally, the preference patterns predicted for the test user are converted back to the rating patterns by reversing the conversion model already computed in step 1.

Unlike most other memory-based approaches, where all inferences are based on the surface rating patterns, our new model is able to convert the observed rating patterns into the underlying preference patterns, which, presumably, are more comparable between different users. The final predicted ratings are obtained by converting the estimated preference patterns back to ratings according to the test user's rating patterns.

2.1 Decoupled modeling of preferences and ratings

To illustrate how we can decouple the preference and rating models, we first use a simple example to explain how to convert

ratings into preference likelihoods, and then give a formal description.

2.1.1 Intuition

Consider an example of two users ('A' and 'B') who have similar "taste" of preferences for 10 items. Suppose user 'A' tends to give all items a higher rating than user 'B', as shown in Table 1, where the items are numbered in the ascending order of preferences.

If we rely on the surface rating patters to compute the similarity of these two users, we may find that they are not very similar, even if we normalize each user's ratings by the user's average rating value as done typically in a traditional memory-based approach.

Table 1: Rating information for user 'A' and 'B'

Item	1	2	3	4	5	6	7	8	9	10
Rating(A)	1	2	3	3	3	4	4	4	4	5
Rating(B)	1	1	2	2	2	3	3	4	4	5

How can we map these surface ratings to somehow more comparable underlying preference likelihood? We note that, user 'A' appear to be more "generous" in assigning ratings, as the average rating of all the 10 times is higher for 'A' than for 'B'. , In particular, 'A' has given four items of a rating of '4', while 'B' has given only two. Intuitively, a rating of '4' should mean stronger preference for user 'B' than for user 'A', since the former has assigned '4' to fewer items. Based on this simple analysis, we can see that the *distribution* of ratings does tell us some information about the true underlying preferences that the user has on the objects. More specifically, two aspects of the rating distribution may influence the likelihood of preference:

1) For a particular rating 'R', if there are a large percentage of objects that are rated less or equal to rating 'R', this rating category has a high chance to be preferred by the user. Rating category '5' is such an example.
2) For a particular rating 'R', if there are a large percentage of objects that are rated exactly in this category, this rating category will have less chance to be preferred by the user. Rating category '4' for user 'A' is such an example.

Table 2: The rating distribution for user 'A' and 'B'

User A	Rating(R)	**1**	**2**	**3**	**4**	**5**
	Pr(Rating=R)	0.1	0.1	0.3	0.4	0.1
	Pr(Rating≤R)	0.1	0.2	0.5	0.9	1
	Pr(Rating≤R)−Pr(Rating=R)/2	0.05	0.15	0.35	0.7	0.95
User B	Rating(R)	**1**	**2**	**3**	**4**	**5**
	Pr(Rating=R)	0.2	0.3	0.2	0.2	0.1
	Pr(Rating≤R)	0.2	0.5	0.7	0.9	1
	Pr(Rating≤R)−Pr(Rating=R)/2	0.1	0.35	0.6	0.8	0.95

To combine these two criterions together, we can use the following 'halfway accumulative distribution' to approximate the likelihood of preference:

$$\Pr(\text{R is preferred}) = \Pr(\text{Rating}\le R) - \Pr(\text{Rating}=R)/2 \quad (1)$$

where R is a rating category. The first term on the right hand side Pr(Rating≤R) implements the idea that a user tends to like items with rating category 'R' if many items are rated under that category. By subtracting the second term Pr(Rating=R)/2 from the first term Pr(Rating≤R), we are able to discount the preference of a rating category if many objects are rated as that category.

Table 2 lists the results for Pr(Rating=R), Pr(Rating≤R) and Pr(Rating≤R)−Pr(Rating=R)/2 for both users. We see that the fact that user 'A' is more "generous" than 'B' is now well captured by the different preference mappings. In particular, a rating of '4' means slightly stronger preference if given by 'B' than if given by 'A'. Table 3 shows the estimated preference likelihoods of all the 10 items for both users. Clearly, the two users now look much more similar than in Table 1.

Table 3: Estimated preferences for user 'A' and 'B'

Item	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Pref(A)	0.05	0.15	0.35	0.35	0.35	0.7	0.7	0.7	0.7	0.95
Pref(B)	0.1	0.1	0.35	0.35	0.35	0.6	0.6	0.8	0.8	0.95

The 'halfway accumulative distribution' shown in Equation (1) thus allows us to convert ratings into the likelihood of preference. For "reverse engineering", i.e., finding the most likely rating category given a computed likelihood of preference, we can simply find the category that is most likely to generate the likelihood of preference. Interestingly, the 'halfway accumulative distribution' in Equation (1) can also be proved by the maximum likelihood estimation. The details of the proof are shown in the Appendix. Furthermore, it is not difficult to show that the averaged preference likelihood of any user over all the rated items, as computed in Equation (1), is exactly 0.5. Therefore, by using the conversion formula in Equation (1), we are able to achieve the similar effect as the Pearson Correlation method, namely adjusting every user to have the same mean in their preference patterns.

2.1.2 Formal Description

Let x and y be an item and a user, respectively. Let $R_y(x)$ stand for the rating that user y gives to item x and $P_y(x)$ stand for the likelihood that user y prefers item x. Then, our task is to find out the distribution of likelihood of preference $P_y(x)$ given the rating category $R_y(x)$. For the sake of simplicity, we can assume that the distribution of likelihood of preference $P_y(x)$ given the rating information $R_y(x)$ is a Gaussian distribution $N(\mu(R_y(x),y), \sigma(R_y(x),y))$, which centers at $\mu(R_y(x),y)$ with width of $\sigma(R_y(x),y)$. Notice that both the mean and width of the distribution depend on not only the rating $R_y(x)$ but also the user y, which indicates that even two different users may give an object the same rating, the distribution of the likelihood of preference on that object can still be different. According to the discussion in the previous subsection, we can use the 'halfway accumulative distribution' of a rating category as the averaged likelihood of preference for that category. Therefore, the mean of the Gaussian $\mu(R_y(x),y)$ can be written as:

$$\mu(R_y(x), y) = \Pr(R \le R_y(x) \mid y) - \Pr(R = R_y(x) \mid y)/2 \quad (2)$$

where $\Pr(R \le R_y(x) \mid y)$ is the likelihood for user y to rate any item no higher than the rating category $R_y(x)$, and $\Pr(R = R_y(x) \mid y)$ is the likelihood for user y to rate any item as category $R_y(x)$. For the width of the Gaussian distribution $\sigma(R_y(x),y)$, we can simply set it as $\Pr(R = R_y(x) \mid y)$, i.e.

$$\sigma(R_y(x), y) = \Pr(R = R_y(x) \mid y) \quad (3)$$

This is reasonable, because, intuitively, the more objects a user give category 'R' to, the higher the variance in the likelihood of being preferred by that user within that rating category 'R' will be. Therefore, for a particular rating category 'R', the variance of preference likelihood should be proportional to the percentage of objects rated within that category, i.e. $\Pr(R=R_y(x)|y)$. With the defined Gaussian distribution $N(\mu(R_y(x),y),\sigma(R_y(x),y))$, we can do two things:

1) *Find out the averaged likelihood of being preferred given the rating category $R_y(x)$.* Since we use Gaussian distribution for describing the distribution of likelihood of being preferred, the averaged preference likelihood given a rating category of a user is simply the mean of the Gaussian distribution $\mu(R_y(x),y)$. To simplify the computation, in the later discussion, we will use the averaged preference likelihood given a rating category to replace the Gaussian distribution.
2) *Compute the most likely rating category given a computed likelihood of being preferred*. To do this, we can simply search for the Gaussian distribution of a rating category that has the highest probability of generating the computed preference likelihood, i.e.

$$R^*(x) = \arg\max_R \frac{1}{\sqrt{2\pi\sigma^2(R,y)}} \exp\left(\frac{(P_y(x)-\mu(R,y))^2}{2\sigma^2(R,y)}\right) \quad (4)$$

2.2 Computing user similarity

As already pointed out before, there are two different notions of similarity between users: similarity in the preference pattern and similarity in the rating pattern. Let $w^P{}_{y,y'}$ stand for the preference similarity between user *y* and user *y'*, and $w^R{}_{y,y'}$ for the rating. We now discuss how to compute them.

2.2.1 Computing rating similarity

We define the similarity of user *y* and *y'* in rating patterns as the likelihood of mistaking user *y* as user *y'* given the rating pattern of *y*. In order to define it rigorously, we first introduce some definitions.

Let {1, 2, …, R_{max}} be the set of rating categories. A rating category *i* is preferred to category *j* if and only if number *i* is larger than number *j*. Let $\tilde{R}(y)$ be user *y*'s rating pattern, which is defined as the set of counts of objects rated as different rating categories, or

$$\tilde{R}(y) = \{C(R,y), 1 \le R \le R_{\max}\} \quad (5)$$

where $C(R, y)$ is the count of objects rated as category *R* by user *y*.

With the above definitions, $w^R{}_{y,y'}$, the rating similarity of user *y* and user *y'*, can be defined as:

$$\begin{aligned} w^R{}_{y,y'} &= p(y'|\tilde{R}(y)) = \frac{p(\tilde{R}(y)|y')p(y')}{p(\tilde{R}(y))} \\ &= \frac{p(\tilde{R}(y)|y')p(y')}{\sum_{y''} p(\tilde{R}(y)|y'')p(y'')} \approx \frac{p(\tilde{R}(y)|y')}{\sum_{y''} p(\tilde{R}(y)|y'')} \end{aligned} \quad (6)$$

The last step is realized by assuming that every user has the same prior $p(y)$. By assuming every rating is generated independently of others, we further express $p(\tilde{R}(y)|y')$ as:

$$p(\tilde{R}(y)|y') = \prod_{R=1}^{R_{max}} \Pr(R|y')^{C(R,y)} \quad (7)$$

Where $\Pr(R|y)$ stands for the likelihood for user *y* to come up with rating *R* for an arbitrary object. By plugging Equation (7) into Equation (6), we are able to compute $w^R{}_{y,y'}$, which is useful for smoothing user patterns.

2.2.2 Compute preference similarity

To compute preference similarity of two users, we first derive the preference pattern of each user using the method described in section 2.1, and then compare the derived preference patterns. According to section 2.1, the likelihood for object *x* to be preferred by user *y*, $P_y(x)$, can be approximated as $\mu(R_y(x),y)$ where $R_y(x)$ is the rating of object *x* by user *y*. Thus, similar to the computation of the rating similarity, the preference similarity of user *y* and user *y'*, $w^P{}_{y,y'}$, can also be computed as the likelihood of mistaking user y' as user y given the preference pattern of *y*.

Let $\tilde{X}(y)$ be the set of objects that are rated by user *y*. The preference pattern of user *y*, $\tilde{P}(y)$, is defined as

$$\tilde{P}(y) = \{P_y(x) | \forall x \in \tilde{X}(y)\} = \{\mu(R_y(x),y) | \forall x \in \tilde{X}(y)\} \quad (8)$$

The preference similarity of user y and user y', $w^P{}_{y,y'}$, can be written as:

$$\begin{aligned} w^P{}_{y,y'} &= p(y'|\tilde{P}(y)) = \frac{p(\tilde{P}(y)|y')p(y')}{p(\tilde{P}(y))} \\ &\approx \frac{p(\tilde{P}(y)|y')}{\sum_{y''} p(\tilde{P}(y)|y'')} \end{aligned} \quad (9)$$

Similarly, we assume that each item is generated independently of others, and express $p(\tilde{P}(y)|y')$ as:

$$\begin{aligned} p(\tilde{P}(y)|y') &= \prod_{x \in \tilde{X}(y)} P_{y'}(x)^{P_y(x)} \\ &= \prod_{x \in \tilde{X}(y)} \mu(R_{y'}(x),y')^{\mu(P_y(x),y)} \end{aligned} \quad (10)$$

There is, however, a flaw in the above expression, i.e. there are items that are rated by user *y* but not rated by user *y'*. In that case, we will not have rating information $R_{y'}(x)$ for user y' on object *x*, which is required in Equation (10). This corresponds to the missing data issue that has been discussed in the introduction section. There are two heuristic solutions:

1) *Item default preference*. Set $P_{y'}(x)$ to the averaged preference likelihood of item *x* over all the users that have rated object *x*.
2) *User default preference*. Set $P_{y'}(x)$ to the averaged likelihood of all objects rated by user y'. This is equivalent to setting $P_{y'}(x)$ to 0.5 uniformly.
3) We will compare the effectiveness of these two methods later.

2.3 Rating Prediction

Ultimately, our goal is to predict the rating for a test user. Let Y be the training user database. For each training user $y \in Y$, the preference pattern is $\tilde{R}_p(y) = \{\Pr(R \mid y) \mid 1 \le R \le R_{\max}\}$, and the rating pattern is

$$\tilde{P}(y) = \{P_y(x) \mid \forall x \in \tilde{X}(y)\} = \{\mu(R_y(x), y) \mid \forall x \in \tilde{X}(y)\}$$

A test user y_0 often has only a small number of items rated. Let $\tilde{X}(y_0)$ stand for the set of items that are rated by user y_0. In order to obtain the rating pattern of the test user y_0, we have two choices:

1) *Simple counting*. Compute $\Pr(R \mid y_0)$ by directly counting the percentage of items rated as category R by user y. However, since there are only a small number of objects rated by user y_0, the number of objects may not be large enough to support a reliable estimate of $\Pr(R \mid y_0)$.

2) *Smoothing with rating patterns of other users*. A better solution to alleviate this sparse data problem is to find out which training users are similar to test user y_0 in rating, and smooth the rating pattern of the test user y_0 with the rating patterns of those similar users. Let $\tilde{R}_p(y_0)$ be the rating pattern obtained by the simple counting method and $\tilde{R}_p'(y_0)$ the smoothed rating pattern using users of similar rating patterns. Then, $\tilde{R}_p'(y_0)$ can be expressed as:

$$\tilde{R}_p'(y) = \{\Pr'(R \mid y_0) \mid 1 \le R \le R_{\max}\}$$

$$\Pr'(R \mid y) = \frac{w^R{}_{y_0,y_0} \Pr(R \mid y_0) + \sum_{y \in Y} w^R{}_{y_0,y} \Pr(R \mid y)}{w^R{}_{y_0,y_0} + \sum_{y \in Y} w^R{}_{y_0,y}} \quad (11)$$

Note that $w^R{}_{y,y_0}$ is used here. With $\tilde{R}_p(y)$ computed for the test user y_0, we can easily compute the preference pattern for user y_0 using Equation (2) and (3). Then, according to section 2.2.2, we can compute the preference similarities of the test user y_0 and any other user, $w^P{}_{y,y_0}$. The expected likelihood for user y_0 to prefer an un-rated item x is then computed as:

$$\hat{P}_{y_0}(x) = \frac{\sum_{y \in Y} w^P{}_{y_0,y} P_y(x)}{\sum_{y \in Y} w^P{}_{y_0,y}} \quad (12)$$

Finally, we predict the rating of an un-rated item x for the user y_0 by converting the expected likelihood of being preferred $\hat{P}_{y_0}(x)$ to the most likely rank $\hat{R}_{y_0}(x)$ using the method described in section 2.1.2.

In summary, there are four steps involved in the prediction of rating for a test user:

1) Compute the rating pattern of the test user by either the simple counting method or smoothing with rating patterns of similar training users. The rate similarity can be computed using the method described in section 2.2.1.

2) Compute the preference pattern of the test user on the rated objects using the method presented in Section 2.1.2.

3) Compute the expected preference likelihood an un-rated item x using Equation (12) with the preference similarity computed using the method described in Section 2.2.2.

4) Predict the rating category for item x by converting the expected preference likelihood to the most likely rating category.

One advantage of this model, compared with the previous approaches to collaborative filtering is the explicit conversion of the rating pattern into the underlying preference pattern for all users. Moreover, both user comparisons and prediction combinations are operated at the level of the preference patterns, instead of the rating patterns.

3. Experiments

We need to answer three questions about this new model for collaborative filtering:

1) *Is the new model for collaborative filtering effective?* To answer this question, we will compare our model with several representative existing approaches. Furthermore, we will vary the size of user database and the number of items that are rated by the test user to see the effectiveness of our model in different situations.

2) *How does the default preference likelihood influence the performance?* In section 2.2.2, in order to compute the preference similarity between users, we have to specify the default preference likelihood of a user for an item that is not rated by that user. We will examine the effectiveness of the three methods proposed in Section 2.2.2.

3) *How does smoothing of rating patterns influence the performance of our model?* In Section 2.3, in order to handle the sparse data problem in computing the rating pattern for a test user, we suggest smoothing the rating pattern of the testing user with the rating patterns of similar users. We would like to examine the effectiveness of this smoothing method.

3.1 Baseline methods

Following [1,2,12], we compare our method to several existing methods, including the Pearson Correlation Coefficient method, the vector similarity method and the Personality diagnosis method. These methods are described below.

- ***Pearson Correlation Coefficient (PCC)***

According to [1], Pearson Correlation Coefficient method predicts the rating of a test user y_0 on item x as:

$$\hat{R}_{y_0}(x) = \overline{R}_{y_0} + \frac{\sum_{y \in Y} w_{y_0,y} (R_y(x) - \overline{R}_y)}{\sum_{y \in Y} w_{y_0,y}} \quad (13)$$

where the coefficient w_{y,y_0} is computed as

$$w_{y_0,y} = \frac{\sum_{x \in \tilde{X}(y) \wedge \tilde{X}(y_o)} (R_y(x) - \bar{R}_y)(R_{y_0}(x) - \bar{R}_{y_0})}{\sqrt{\sum_{x \in \tilde{X}(y) \wedge \tilde{X}(y_o)} (R_y(x) - \bar{R}_y)^2} \sqrt{\sum_{x \in \tilde{X}(y) \wedge \tilde{X}(y_o)} (R_y(x) - \bar{R}_y)^2}} \quad (14)$$

- ***Vector Similarity (VS)***

This method is very similar to the previous method except that the correlation coefficient w_{y,y_0} is computed as:

$$w_{y_0,y} = \frac{\sum_{x \in \tilde{X}(y) \wedge \tilde{X}(y_o)} R_y(x) R_{y_0}(x)}{\sqrt{\sum_{x \in \tilde{X}(y)} R_y(x)^2} \sqrt{\sum_{x \in \tilde{X}(y_0)} R_{y_0}(x)^2}} \quad (15)$$

- ***Personality Diagnosis (PD)***

This is a method introduced by Pennock et al. in [4]. It treats each user in the training pool as an individual model. In order to compute the ratings for active (test) users, it first computes the likelihood of generating the observed ratings of the active user for each model (e.g. user) within the training pool, i.e.

$$p(R_{y_0} \mid R_y) \propto \prod_{x \in \tilde{X}(y_0)} \exp\left(-\frac{(R_y(x) - R_{y_0}(x))^2}{2\sigma^2}\right) \quad (16)$$

Then, the likelihood for the test user to rate an unrated item *x* as *v* is computed as:

$$p(R_{y_0}(x) = v \mid \{R_y\}) \propto \sum_y p(R_{y_0} \mid R_y) \exp\left(-\frac{(R_y(x) - v)^2}{2\sigma^2}\right) \quad (17)$$

In our experiments, the standard deviation σ is set to 1

3.2 Description of Datasets

We experimented with two commonly used data sets of movie ratings. The first is named 'MovieRating', which can be obtained from the web site [9], and the second one is a subset of the standard EachMovie dataset obtained from the web site of Compaq research [10]. For the EachMovie dataset, 2000 users were randomly selected from those who have rated more than 40 movies. We did not use all the available user data because we want to test our algorithms in a more challenging and realistic scenario when the system has not yet acquired a large user base. In general, we should expect an algorithm to perform better with more user data. The basic properties of these two datasets are listed in Table 3.

Table 3: Description of experimental datasets

	MovieRating	EachMovie
Number of Ratings	5	6
Number of Users	500	2000
Number of Items	1000	1648
Average number of rated items	87.73	129.62

To compare the methods in full spectrum, we vary both the number of users in the training database and the number of observed items that are rated by the test user, as what have been done in [1, 8, 12]. For the smaller database 'MovieRating', we allow the number of users in the training database to be 100 and 200, and the number of exposed items rated by the test users to be 5, 10 and 20. For the larger database 'EachMovie', we choose the size of training database to be 200 and 400, and the number of exposed items that are rated by the test users to be 5, 10 and 20. In all cases, the rest of the users are used for testing. The reason for choosing a relatively larger number of users in the training database for the 'EachMovie' database is because the 'EachMovie' database contains considerably larger number of users and movies than the 'MovieRating' database.

3.3 Evaluation Measure

For evaluation, we look at the mean absolute deviation of the predicted ratings from the actual ratings on items that users in the test set have actually rated, i.e.

$$S_{y_0} = \frac{1}{m_{y_0}} \sum_{x \in \tilde{X}(y_0)} |R_{y_0}(x) - \hat{R}_{y_0}(x)| \quad (15)$$

where $\hat{R}_{y_0}(x)$ is the predicted rating on item *x* by user y_0, $R_{y_0}(x)$ is the actual rating on item *x* by user y_0 and m_{y0} is the number of test items that have been rated by the test user y_0. We refer to this measure as the mean absolute error (MAE) in the rest of this paper. There are some other measures like the Receiver Operating Characteristic (ROC) as a decision-surrport accuracy measure [7] and the normalized MAE. But since MAE is the most commonly used metric and has been reported in most previous research [1, 2, 7, 8, 12], we chose it as the evaluation measure in our experiments to make our results more comparable.

3.4 Experiment Results

In order to answer the three questions raised at the beginning of this section, we design three corresponding groups of experiments. The results of these three experiments are discussed in the following three subsections.

3.4.1 Effectiveness of the New Mode

We compare the new model with the three baseline methods over the two datasets using the MAE as the evaluation metric. The results are shown in Table 4 and 5, respectively.

From Table 4 and 5, we see that the new model outperforms all the four baseline methods substantially and consistently over both the 'MovieRating' and 'EachMovie' databases and for all the settings. The improvement is substantial because, over both the 'MovieRating' and 'EachMovie' databases, the new model with the least information (i.e. the smallest number of rated items and smallest number of users in the training set) performs even better than all the baseline methods with the most information. Therefore, we can conclude that the new model is more effective in predicting the ratings for the test users than the existing methods we tested.

3.4.2 Effectiveness of Smoothing in Computing the Rating Pattern

In Section 2.3, we present two different ways to compute the rating pattern for the test user, either based on the unsmoothed percentage of each rating category or through smoothing the rating pattern of the test user with other similar users in the training database. In Table 6 and 7, we show the performance of

Table 4: MAE results for four baseline methods and the new model on the 'MovieRating' database. A smaller number indicates a better performance.

		5 Items rated	10 Items rated	20 Items rated
100 Training Users	PCC	0.881	0.832	0.809
	VS	0.859	0.834	0.823
	PD	0.839	0.826	0.818
	New Model	**0.788**	**0.776**	**0.763**
200 Training Users	PCC	0.878	0.828	0.801
	VS	0.862	0.950	0.854
	PD	0.835	0.816	0.806
	New Model	**0.768**	**0.760**	**0.743**

Table 5: MAE results for four baseline methods and the new model on the 'EachMovie' database. A smaller number indicates a better performance.

		5 Items rated	10 Items rated	20 Items rated
200 Training Users	PCC	1.22	1.16	1.13
	VS	1.25	1.24	1.26
	PD	1.19	1.16	1.15
	New Model	**1.04**	**1.01**	**0.98**
400 Training Users	PCC	1.22	1.16	1.13
	VS	1.32	1.33	1.37
	PD	1.18	1.16	1.15
	New Model	**1.03**	**1.00**	**0.97**

Table 6: MAE results for the new model on the 'MovieRating' database with and without smoothing in computing the rating pattern for test users. A smaller number indicates a better performance.

		5 Items	10 Items	20 Items
100 Training Users	w.o. smoothing	0.814	0.788	0.771
	w.i. smoothing	0.788	0.776	0.763
200 Training Users	w.o. smoothing	0.789	0.766	0.747
	w.i. smoothing	0.768	0.760	0.743

Table 7: MAE results for the new model on the 'EachMovie' database with and without smoothing in computing rating pattern for testing users. A smaller number indicates a better performance.

		5 Items	10 Items	20 Items
200 Training Users	w.o. smoothing	1.06	1.01	0.98
	w.i. smoothing	1.04	1.01	0.98
400 Training Users	w.o. smoothing	1.05	1.01	0.98
	w.i. smoothing	1.03	1.00	0.97

the new model with and without the smoothing procedure in computing the rating pattern for the test user.

In general, smoothing of the rating pattern outperforms the unsmoothed simple percentage. Particularly, when the number of exposed items rated by the test user is small, the smoothing technique appears to help even more in the performance. Take the 'MovieRating' database with 100 training users as an example. When there are only 5 rated items exposed to the system, with the smoothing technique, the MAE is improved by 0.026. When the number of exposed items rated by the test user grows to 10 and 20, the improvement in MAE decreases to 0.012 and 0.008. This is expected, since when the number of rated items of test users is smaller, the sparse data issue is more severe in estimating the rating pattern. Therefore, the smoothing technique is more helpful when there are only a small number of rated items available.

3.4.3 Effectiveness of Different Default Preferences

In section 2.2.2, we discussed the issue of missing data, namely the items that are rated by the test user but not by the training users. We proposed two different strategies for filling out the missing preferences for these items, i.e., item default preference and user default preference.

Table 8: MAE results for the new model on the 'MovieRating' database using the two default preference-setting methods. A smaller number indicates a better performance.

		5 Items	10 Items	20 Items
100 Training Users	Item default	0.787	0.777	0.765
	User default	0.787	0.777	0.762
200 Training Users	Item default	0.768	0.762	0.742
	User default	0.768	0.761	0.742

Table 9: MAE results for the new model on the 'EachMovie' database using the two default preference-setting methods. A smaller number indicates a better performance.

		5 Items	10 Items	20 Items
200 Training Users	Item Default	1.04	1.01	0.98
	User Default	1.04	1.01	0.98
400 Training Users	Item Default	1.03	1.00	0.98
	User Default	1.03	1.00	0.97

These two methods are compared in Table 8 and Table 9. Interestingly, the two strategies have almost identical performance in all cases. We suspect that the mean preference likelihood by averaging the preference of different users on the same item may actually be around 0.5, which is the average preference value of a user.

4. Conclusions and Future Work

In this paper, we propose a new probabilistic framework for collaborative filtering. The model effectively addresses a critical issue in collaborative filtering, that is, users with similar preference patterns can rate objects differently due to the use of different rating schemes. Unlike previous work, where user similarities are computed directly using the surface rating information, our model first converts the surface ratings of objects into an underlying preference likelihood, and then computes the

similarity between users based on the derived preference patterns instead of the surface rating patterns. By doing so, we are able to explicitly separate the preference information from the rating information. More specifically, we have two models for each user. One is a preference model, capturing which type of objects a user prefers, and the other is a rating model, capturing how a user rates objects according to his/her preference. Our experiments on two commonly used movie rating data sets show that the proposed new model consistently and substantially outperforms four representative existing methods.

As stated in Section 2.2.1, when a rating is converted into the likelihood of preference, it should result in a distribution. However, for the simplicity of computation, we simply use the mean of the distribution to replace the whole distribution. One future research direction would be to compute the similarity of users directly using the distribution instead of the mean of the distribution. Another limitation of this work is that the transformation of ratings into the likelihood of preference is defined according to a very simple probabilistic model. A better solution would be to find out the transformation according to more sophisticated optimization criterion, which is another interesting direction to further explore. For example, a graphic model may be used to find the optimal transformation.

5. REFERENCES

[1] J. S. Breese, D. Heckerman and C. Kadie, Empirical Analysis of Predictive Algorithms for Collaborative Filtering, Proceeding of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI), 1998.

[2] J. L. Herlocker, J. A. Konstan, A. Brochers and J. Riedl. An Algorithm Framework for Performing Collaborative Filtering. In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), 1999.

[3] I. M. Soboroff and C. Nicholas. Collaborative Filtering and the Generalized Vector Space Model. In Proceedings of the 23rd Annual International Conference on Researech and Development in Information Retrieval (SIGIR), 2000.

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

[5] T. Hofmann and J. Puzicha, Latent Class Models for Collaborative Filtering, In Proceedings of International Joint Conference on Artificial Intelligence (UAI), 1999.

[6] I. M. Soboroff and Charles K. Nicholas. Combining Content and Collaboration in Text Filtering. In Proceedings of the IJCAI'99 Workshop on Machine Learning for Information Filtering, 1999

[7] P. Melville, R. J. Mooney, R. Nagarajan, Content-Boosted Collaborative Filtering for Improved Recommendations. In Proceedings of the the Eighteenth National Conference on Artificial Intelligence (AAAI), 2002.

[8] SWAMI: a framework for collaborative filtering algorithm development and evaluation. In Proceedings of the 23rd Annual International Conference on Researech and Development in Information Retrieval (SIGIR), 2000.

[9] http://www.cs.usyd.edu.au/~irena/movie_data.zip

[10] http://research.compaq.com/SRC/eachmovie

[11] A. Dempster, N. Laird and D. Rubin. Maximum Likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, B 39:1-38, 1977.

[12] D. M. Pennock, E. Horvitz, S. Lawrence and C. L. Giles, Collaborative Filtering by Personality Diagnosis: A Hybrid Memory- and Model-Based Approach, in Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI), 2000.

6. Appendix

In this appendix, we will derive Eqn. (1) by using Maximum Likelihood Estimation (MLE). Let $\{1, 2, \ldots, R_{max}\}$ be the set of rating categories, $\{X_1, X_2, \ldots, X_{Rmax}\}$ be the set of items, each being assigned a rating category, and $\theta=\{p_1, p_2, \ldots, p_{Rmax}\}$ be the set of probabilities for each category to be preferred. Assuming that L is a ranking list that is consistent with the rating pattern, namely all the items that rated in a high category will be put at the beginning of the ranking list. The appropriate preference probabilities θ should have a large likelihood to generate such consistent ranking list, namely

$$\theta^* = \arg\max_{\theta} \Pr(L \mid \theta) \tag{A1}$$

The generation probability for the ranking list L $\Pr(L|\theta)$ can be written as the product of likelihood of generating the correct order for any two items, i.e.

$$\Pr(L \mid \theta) = \prod_{i<j} \left(\prod_{\substack{x \in X_j \\ x' \in X_i}} \Pr(x > x' \mid \theta) \right) \prod_i \prod_{x \neq x' \in X_i} \Pr(x > x' \mid \theta) \tag{A2}$$

Where, $\Pr(x>x')$ is the probability that item x is preferred to item x'. Notice that for two items belonging to the same category, any order will be consistent with the rating patterns. $\Pr(x>x')$ can be written as expressions of preference probabilities θ, i.e.:

$$\Pr(x > x' \mid \theta) = p_{R(x)}(1 - p_{R(x')}) \tag{A3}$$

With expression (A3) and (A2), we have the likelihood $\Pr(L|\theta)$ as:

$$\Pr(L \mid \theta) = \left\{ \begin{array}{l} \prod_{R<R'} (1-p_R)^{C(R,y)} p_{R'}^{C(R',y)} \times \\ \prod_R \left((1-p_R) p_R \right)^{C(R,y)(C(R,y)-1)/2} \end{array} \right\} \tag{A4}$$

By maximizing the likelihood $\Pr(L|\theta)$, we can compute the most likely preference probabilities θ, which will directly result in the Equation (1).