ACKNOWLEDGMENTS

I would like to take this opportunity to thank Dr. Jonathan L. Herlocker for all of his support, ideas, and positive input throughout the research and writing phases of this report. He has been instrumental in my education.

I also owe Dr. Thomas G. Dietterich a debt of gratitude for all of his help for providing theoretical and applied information based upon his vast experience within this field.

I would also like to thank the other member of my committee, Dr. Bella Bose, for his valuable comments and his time.

I also thank Dr. Margaret M. Burnett for her support and encouragement.

I finally would like to thank my husband for providing support throughout this project. Lastly, I want to thank my family back home, especially my mother, for their support throughout my college education.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Collaborative Filtering with Probability and Collaborative Filtering with Privacy

1. INTRODUCTION

The Internet presents us with an explosion of new, mostly computer-based, information. According to [Lyman and Varian, 2000], it is estimated that the world produces between 1 and 2 exabytes of unique information per year, or roughly 250 megabytes for every man, woman and child on earth. This is equivalent to the textual content of 250 books. It is far from a trivial task to get the right information at the right time for a specific user. This difficulty has been termed information overload. A world-wide survey [Waddington, 1996] found that one third of managers suffer from ill-health because of information overload. The term "Information Fatigue Syndrome" is proposed to describe the resulting symptoms.

Obviously, not all of the exploding information is equally important to each user. The user should separate the wheat from the chaff. To help users sift through the large quantities of online information and locate the relevant material, two most common approaches, content-based and collaborative filtering, are proposed.

The content-based filtering system locates the information for a specific user based on the queries, containing words that the user is interested in, provided by that user. The current search engines, such as Google, fall into this category. In spite of the success of Google, the content-based filtering system has the following limits:

1. While the system is well suited to locating textual documents (e.g. Web pages) relevant to a topic, it does not cater for categories of non-textual information, such as music or movies.

2. The system cannot filter information based on quality or taste. For example, it cannot distinguish a well-written article from a badly written one.

3. The system suffers from the problems of polysemy and synonmyny, i.e., it is difficult to distinguish between words that are spelled the same way but have different interpretations or spelled differently but mean the same thing.

The collaborative filtering system can overcome the above limits, so it is quickly becoming a popular technique for reducing information overload, often as a technique to complement the content-based filtering system. The term "collaborative filtering" was coined by Goldberg et al. [Goldberg et al., 1992], and it is based on the underlying assumption that human preferences are correlated [Pennock et al., 2000]. Collaborative filtering systems predict the preferences of a user, so called the active user, based on the recorded preferences of that user and other like-minded users.

The basic idea of collaborative filtering has been widely and unconsciously used in our daily life. We often recommend the movies we like to our like-minded friends and we also often receive the recommendations from them. While these recommendations are restricted among you and your known friends, collaborative filtering systems can make recommendations based on the preferences of a much larger community of people, most of them you may never meet. Thus collaborative filtering systems can provide more reliable recommendations for the users by finding preferences of more like-minded neighbors.

The quick growth of e-commerce presents another challenge and opportunity to collaborative filtering. Imaging that in the good old days, whenever you walked into a local bookstore, you were immediately recognized and greeted by the shop owner and guided to new releases she knew would interest you. But when you visit

a behemoth Web site to make a purchase, mostly often, you are totally lost in the huge collections. [Parkes, 2001] For years, e-tailers have tried to remedy this problem by using the collaborative filtering system. In a simple case, if you shopped on Amazon.com before, you'll automatically be shown the recommendations, such as "customers who bought items in your recent history also bought", the next time you visit there. As collaborative filtering has seen considerable success on the Internet, some strong products have been offered in the market.

For example, Oracle releases Oracle9*i* Personalization as an option to Oracle9*i* Application Server, which provides a way for a Web site to customize or personalize the recommendations it presents to Web site visitor. Oracle9*i* Personalization uses two recommendation algorithms to create models, predictive association rules and transactional Naïve Bayes. Both algorithms are based on a theorem of Bayes concerning conditional probability. [Oracle, 2001]

IBM offers WebSphere Personalization, which runs on top of IBM's WebSphere Application Server. It's powered by two main personalization engines: a rules engine, which executes a set of business rules that determine which Web content is displayed for a particular user; and a recommendation engine, which supports content and product recommendations via Macromedia's LikeMinds patented real-time collaborative filtering technology. In the later case, the server first builds a profile comprised of potentially thousands of preference and behavior data points for each Web visitor, and then this profile is used to create an affinity group of like-minded visitors and generate individualized recommendations for each visitor. [Greening, 2000]

Collaborative filtering technology developed at Microsoft Research [Breese et al., 1998] [Heckerman et al., 2000], including algorithms for prediction and segmentation based on Bayesian statistics, has also been shipped in Microsoft SQL Server 2000 and Commerce Server 2000 as well as the WinMine Toolkit. In one case, the system is to guide you to the Web pages others with similar interests have

selected. The system starts with comparing the pages you are viewing to a database of pages that anonymous visitors have viewed in the past, and then it finds the anonymous visitors who have visited similar pages to you and recommends other pages that they have visited.

Meanwhile, there are several emerging data mining, a broad concept about collaborative filtering, standards, including the Object Management Group's Common Warehouse Metadata (CWM), the Data Mining Group's Predictive Mining Markup Language (PMML), the Java Community Process's Java Data Mining (JDM) and International Standards Organization's SQL/MM for Data Mining.

Although collaborative filtering has contributed significantly to the areas related to information overload and e-commerce, the current developed systems are flawed in a number of respects.

1. Privacy

A user can benefit from collaborative filtering after she provides her preferences. At the same time, it introduces the danger that the user's opinion may be maliciously used and thus raises a serious privacy issue. A detailed discussion can be found in the later section.

2. Sparsity

In many domains, the number of ratings the users can provide is much less than the total number of items. If we represent the users' ratings on items using a matrix, each entry, $v_{i,j}$, representing the *i*-th user's rating on the *j*-th item, i.e., each column for an item and each row for a user, it is very sparse. The sparsity problem makes it very difficult to predict an item on which few ratings are available. Furthermore, only based on a few ratings, the prediction will be rather inaccurate.

3. Scalability

Most collaborative filtering algorithms require computation that grows with both the number of users and the number of items. With millions of users and items, these algorithms will suffer serious scalability problems.

4. Bootstrap

All collaborative filtering systems experience the "cold start" problem: one needs ratings to predict ratings. Virtual users may be synthesized during the initial phase of a recommender system, who represent the specific types of users.

In this project, we present two new approaches to address some of the above issues. The project is organized into two parts.

In the first part, we introduce a new algorithm, a distribution-based algorithm. The experimental results show that this new algorithm can maintain the accuracy when the dataset is sparse, and provide the probabilistic recommendations, which can satisfy the different needs for the various users.

In the second part, we discuss a new "blurring profile" solution to protect privacy in detail. The experimental results show that our new solution can protect users' privacy as well as maintain reasonable accuracy, in some cases even increase accuracy.

2. COLLABORATIVE FILTERING WITH PROBABILITY

The goal of collaborative filtering is to predict the preferences of one user, the active user, based on the preferences of a group of users. Up to now, the nearest-neighbor method is one of the most successful collaborative filtering algorithms. The key idea is to find the neighbors with similar taste, ideally like-minded neighbors, and then recommend the items the neighbors prefer. Although the nearest-neighbor method shows reasonable performance and computational complexity [Herlocker et al., 1999], it suffers from several drawbacks. The sparse rating problem is one of them, i.e., the number of ratings obtained from a specific user is much less than the number of total items. Moreover, the number of ratings used in predicting an item may dramatically differ from one prediction to another. Intuitively, we should recommend an item with 100 neighbors' good ratings rather than an item with only one neighbor's good rating. However, the current nearest-neighbor method treats both recommendations equally.

In order to overcome this shortcoming, a new algorithm, a distribution-based algorithm, is proposed here, which provides a probability for each predicted rating. Basically, it gives more confidence to items that have more positive ratings. This new algorithm is also designed to answer different questions and satisfy different needs. For example, it could be used to recommend truly excellent items or not bad items, or point out the bad items to remove them from a list, or point out the extreme ratings for research purpose. Similar to the traditional nearest-neighbor method, this algorithm is also easy to understand and easy to implement.

This chapter is organized as follows. Firstly, we briefly review some collaborative filtering algorithms related to our work. Secondly, we explain the proposed distribution-based algorithm in detail. Then, we evaluate its performance on the EachMovie dataset. The results show that our new algorithm significantly outperforms the previous nearest neighbor method for small user profile or sparse training set.

2.1 LITERATURE REVIEW

In this section, we briefly review some collaborative filtering algorithms related to our work.

The collaborative filtering algorithm is the core for making predictions. Initially the algorithm came from the statistical theory, and recently it is cast as a machine-learning problem. [Breese et al., 1998] identifies two major classes of prediction algorithms: memory-based and model-based algorithms.

Memory-based algorithms maintain a database of all users' known preferences for all items, and, for each prediction, perform some computation across the entire database. The nearest-neighbor method is the most successful memory-based algorithm, which generates predictions by weighting each other user's ratings proportionally to his or her similarity to the active user. A variety of similarity metrics are proposed, including the Pearson correlation coefficient and vector similarity based on the vector cosine measure. Recently, [Yu et al., 2001] applies mutual information based feature weighting to modify the Pearson correlation coefficient. The similarity measure, $w_{a,i}$, between the active user *a* and the user *i* is shown as follows:

$$w_{a,i} = \frac{\sum_j w_j^2 \left(v_{a,j} - \bar{v}_a\right)\left(v_{i,j} - \bar{v}_i\right)}{\sqrt{\sum_j w_j^2 \left(v_{a,j} - \bar{v}_a\right)^2 \sum_j w_j^2 \left(v_{i,j} - \bar{v}_i\right)^2}} \tag{2-1}$$

where $v_{a,j}$ and $v_{i,j}$ are the votes cast by users *a* and *i* on item *j*, respectively, $\bar{v}_a$ and $\bar{v}_i$ are the mean votes for users *a* and *i*, respectively. w_j represents the weight of item *j* with respect to the target item, which is treated as the mutual information, a dependency measure, between item *j* and target item *t*.

$$MI\left(V_j; V_t\right) = H\left(V_j\right) + H\left(V_t\right) - H\left(V_j, V_t\right) \tag{2-2}$$

where V_j and V_t are the votes on item *j* and target item *t* respectively, $H(V_j)$ and $H(V_t)$ are the entropies of V_j and V_t, and $H(V_j,V_t)$ is the joint entropy between two items. The entropy *H(S)* of a random variable *S* is defined as follows:

$$H(S) = -\int_{\Omega_S} f_S(x)\log_2 f_S(x)dx \tag{2-3}$$

where Ω_S is the domain of *S* and $f_S(x)$ is the density function of *S*. *H(S)* is a measure of uncertainty inherent in the value of *S*.

As observed in [Yu et al., 2001], the mutual information based feature weighting could further improve the accuracy by about 4 ~ 5% with respect to the Mean Absolute Error (MAE).

Model-based algorithms use the user's preference database to learn a model, which is then used for predictions. The model, which could be built off-line over several hours or days, is very small and very fast. Model-based algorithms are suitable for the environments in which users' preferences change slowly compared to the time needed to update the model.

[Breese et al., 1998] describes and evaluates two probabilistic models, the Bayesian clustering and Bayesian network models. In the Bayesian clustering model, like-minded users are clustered together into classes. Given her class membership, a user's ratings regarding the various items are assumed to be independent. The parameters of the model are estimated from the user database using the EM (Expectation Maximization) algorithm, and the number of classes is chosen by selecting the model structure that yields the largest marginal likelihood of the data. In the Bayesian network model, a decision tree is built for each item from a training set of user preferences. The structure of the tree encodes the dependencies between items and the leaves represent the conditional probabilities based on a set of parent items that are the best predictors of its preferences.

[Pennock et al., 2000] describes and evaluates a hybrid memory- and model-based method called personality diagnosis (PD). The idea is that given the active user's known preferences for some items, the method computes the probability that she is of the same "personality type" as every other user, and, in turn, the probability that she will like some new items. The method is summarized as follows, since it is similar to the one we propose here.

Let n be the number of users and m the total number of items. Denote the $n \times m$ matrix of all users' ratings for all items as $\boldsymbol{R}$. The rating of user i for item j is R_{ij} and $\boldsymbol{R_i}$ is the ith row of $\boldsymbol{R}$, or the vector of all of user i's ratings. The method assumes that user i's personality type can be described as a vector $R_i^{true} = \left\langle R_{i1}^{true}, R_{i2}^{true}, ..., R_{im}^{true} \right\rangle$ of "true" ratings for all seen titles. It is assumed that users report ratings for items they've seen with Gaussian noise, i.e.,

$$\Pr\left(R_{ij} = x \middle| R_{ij}^{true} = y\right) \propto e^{-(x-y)^2/2\sigma^2} \tag{2-4}$$

where σ is a free parameter.

It is also assumed that

$$\Pr\left(R_a^{true} = R_i\right) = \frac{1}{n} \tag{2-5}$$

Given the active user's ratings, the probability that the active user is of the same personality type as any other user can be computed as

$$\begin{aligned} &\Pr\left(R_a^{true} = R_i \middle| R_{a1} = x_1, ..., R_{am} = x_m\right) \propto \\ &\Pr\left(R_{a1} = x_1 \middle| R_{a1}^{true} = R_{i1}\right) \cdots \Pr\left(R_{am} = x_m \middle| R_{am}^{true} = R_{im}\right) \Pr\left(R_a^{true} = R_i\right) \end{aligned} \tag{2-6}$$

Then, a probability distribution for the active user's rating of an unseen title *j* can be computed as

$$\Pr\left(R_{aj} = x_j \middle| R_{a1} = x_1, ..., R_{am} = x_m\right) = \sum_{i=1}^{n} \Pr\left(R_{aj} = x_j \middle| R_a^{true} = R_i\right) \Pr\left(R_a^{true} = R_i \middle| R_{a1} = x_1, ..., R_{am} = x_m\right) \quad (2\text{-}7)$$

Finally, the most probable rating is returned as the prediction.

Since PD actually is sort of the Bayesian clustering method with exactly one user per cluster, it has an intrinsic drawback. In a clustering method, the users or items are clustered together into (hopefully meaningful) classes sharing some similar properties. In contrast, PD treats every single user as a distinct class - some users may actually belong to one class - which results in that it is hard to interpret and compute Eq. (2-6).

According to Eq. (2-7), PD can also be though of as the nearest-neighbor method. Instead of directly predicting the rating, PD computes the probability of rating values for an unseen item using the weighted sum of the neighbors' probabilities rating the same values for that target item as the active user. The weight, measuring the similarity between the active user and the neighbor, is the product of the possibilities the neighbor rating the same value as the active user for each item. This potentially requires a relatively flat distribution describing users' rating behavior, otherwise, a small discrepancy between the neighbor's and active user's rating values will lead to a big difference in the resulting weight. This could be confirmed by the reported experimental results in [Pennock et al., 2000]. For the EachMovie data, the parameter σ is tuned to 2.5, and the relevant users' rating behavior is shown in Figure 2-1.

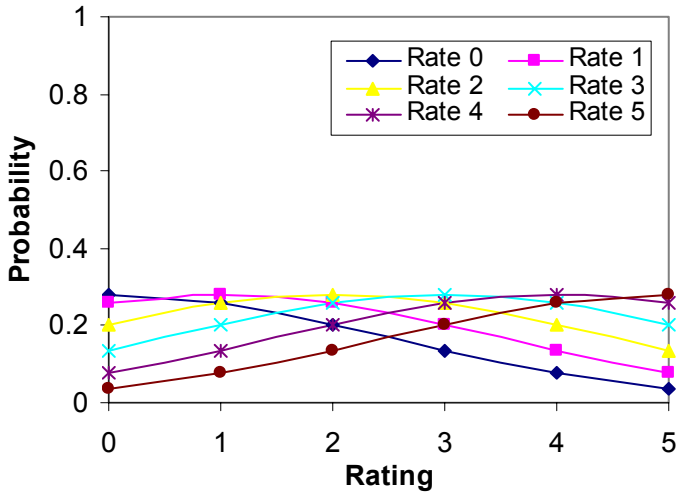

Figure 2-1 The users' rating behavior

The user dataset, basically the matrix $\boldsymbol{R}$ with entry R_{ij} representing the rating of user i for item j, may contain millions of users and items. Many approaches are proposed to reduce its size. Essentially, all the model-based methods belong to these approaches in the sense that they make predictions based on the learned smaller and faster models instead of the original dataset. Some other approaches are summarized here.

There are two straightforward approaches to reduce the user dataset size while still give good predictions. Incremental deletion loops through the dataset, tests each user to see if she can be correctly predicted given the other users, and deletes her from the dataset if so. Incremental growth starts with an empty dataset and adds each user to the dataset only if she cannot be correctly predicted by the users already stored.

[Yu et al., 2001] proposes an instance (user) selection method based on the user's strength of the description. Given a training user i and her rated item set D_i

as well as the corresponding votes, the user's strength with respect to the target item *t* is defined as:

$$S_{i,V_t,D_i} = \frac{\sum_{j \in D_i, j \neq t} MI(V_j, V_t)}{|D_i| - 1} \tag{2-8}$$

where V_j and V_t are the votes on item *j* and target item *t* respectively, and $MI(V_j, V_t)$ is the mutual information between two items. The instance/user selection process starts with, for each target item, computing the strength for all the users who have voted on the target item, and then sorts them in descending order of the strength and selects the top users from the list according to a sampling rate *r*. For each target item, an index table of selected training users is created. As observed in [Yu et al., 2001], an optimal selection rate of 12.5% improves the accuracy by about 4% with respect to MAE.

[Pennock et al., 2000] explores the use of the expected value of information (VOI) to compress the amount of data with as little impact on accuracy as possible. VOI computation identifies, via a cost-benefit analysis, the most valuable new information to acquire in the context of a current probability distribution over states of interest. The approach computes the average information gain of items and/or users in the data set and eliminates those of low value accordingly.

Another set of approaches to reduce the user dataset size includes Singular Value Decomposition (SVD), Principal Component Analysis (PCA) and Factor Analysis (FA), which are the statistical methods for condensing the information contained in the original dataset into a smaller set of dimensions with a minimum loss of information.

SVD [Sarwar et al., 2000] is a matrix factorization technique that factors an $m \times n$ matrix $\boldsymbol{R}$ into three matrices as the following:

$$R = U \cdot S \cdot V' \tag{2-9}$$

where, $\boldsymbol{U}$ and $\boldsymbol{V}$ are two orthogonal matrices of size $m \times r$ and $n \times r$, respectively; $\boldsymbol{S}$ is a diagonal matrix of size $r \times r$ having all singular values of matrix $\boldsymbol{R}$ as its diagonal entries. All the entries of matrix $\boldsymbol{S}$ are positive and stored in decreasing order of their magnitude. SVD provides the best lower rank approximations of the original matrix $\boldsymbol{R}$, in terms of Frobenius norm (square-root of sum of squares of matrix elements). If the $r \times r$ matrix $\boldsymbol{S}$ is reduced to have only k largest diagonal values, and the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are reduced accordingly. The reconstructed matrix $R_k = U_k \cdot S_k \cdot V_k'$ is the closest rank-k matrix to $\boldsymbol{R}$. Obviously, the optimal choice of the value k is critical to high-quality prediction generation.

Instead of factoring the original matrix, PCA [Goldberg et al., 2001] factors its covariance matrix or correlation matrix. The generated eigenvectors are called the principal component axes, which are lines that minimize the average squared distance to each point in the dataset. The principal components are the projections of the original dataset onto the principal component axes. In general, there are as many principal components as items. However, because of the way they are calculated, it is usually possible to consider only a few of the principal components corresponding to the largest eigenvalues, which together explain most of the original variation. To decide how many principal components to retain, a common criterion is to include just enough components to explain some arbitrary amount (typically, 90%) of the variance.

Factor analysis [Canny, 2002] is an attempt to explain the correlations between observable variables in terms of underlying factors, which are themselves not directly observable. Formally, let $\boldsymbol{x}$ be a random variable, representing a user's preferences for p items, with mean μ and covariance matrix Σ, then the k-factor model holds for $\boldsymbol{x}$ if $\boldsymbol{x}$ can be written in the form [S-Plus, 2001]

$$x = \mu + \Lambda f + u \tag{2-10}$$

where $\Lambda = \{\lambda_{ij}\}$ is a $p \times k$ matrix. $\boldsymbol{f}$ and $\boldsymbol{u}$ are random vectors representing, respectively, the k underlying common factors and p unique factors associated with the original observed variables. Equivalently,

$$\Sigma = \Lambda\Lambda' + \Psi \tag{2-11}$$

where $\Psi = VAR(u)$. Again, choosing the number of factors is problematic.

As mentioned, the user dataset, R, is extremely sparse. The EachMovie dataset described later, which contains only 3% of the possible ratings, is considered a "dense" dataset. Other datasets may have 0.1% or even 0.01% of the possible ratings. A usual approach to dealing with sparseness is to replace missing elements with the average of some non-missing elements, either the per-user (row) average, or the per-item (column) average, or the average over all ratings. However, as stated in [Canny, 2002], this approach is simply incorrect from a statistical point of view. The available user ratings induce a probability distribution for each missing rating, which cannot be represented by any single value. One must effectively work with the distributions of missing ratings, not just their expectations. This idea is adopted in our approach to protect the users' privacy described later.

Eigentaste [Goldberg et al., 2001] profiles user taste with universal queries: each user is presented with the same gauge set of items to rate during its profiling phase. This approach is suitable for the domain such as recommending joke, since it may take only a few minutes to read new jokes and give them ratings, while it may take several hours to watch new movies. The ratings may introduce quite biases if only based on a description about items. Also, how to efficiently incorporate the new items into the gauge set is questionable.

The approach used in [Canny, 2002] is to compute symbolic expected values for all the missing quantities including quantities derived from missing ratings, and

to substitute them in the formulas so they are complete. It turns out that substituting for quantities derived from missing ratings causes them to cancel later in the analysis. In the other words, missing values are treated as if they were never there. Specifically, an $n \times n$ "trimming" matrix $\boldsymbol{I}_j$ is introduced, which is a diagonal matrix with 1 in positions (i, i) where i is an item that user j has rated and zero everywhere else. The matrix $\boldsymbol{I}_j$ restricts the formulas for user j to the items they have rated. This approach is effective since the algorithm, Factor Analysis, is implemented using an EM recurrence and so it can be adapted to any subset of the original dataset. Its effectiveness with other collaborative filtering algorithms is unclear.

2.2 DISTRIBUTION-BASED ALGORITHM

In this section, we propose a distribution-based collaborative filtering algorithm.

As mentioned, the dataset the recommender system based on is usually sparse, and the average number of ratings for each item per user has a large variance. Sometimes the recommender system predicts the unexpected items. For example, suppose there is only one rating for the movie "Jeepers Creepers", and it is excellent: 5 (rating scale from 0 to 5). On the other hand, there is another movie "Scream" with 100 ratings, 90 of which are 5 and 10 are 4. Which movie would you rather take your chances with? Obviously, "Scream" is the answer for most people.

However, the current algorithms may predict the movie "Jeepers Creepers" with rating 5 since that rating is the only available information, and may predict the movie "Scream" with a rating between 4 and 5. If the system ranks movies based on the predicted ratings, it will recommend "Jeepers Creepers" rather than "Scream".

To address this issue, we propose a distribution-based algorithm by associating the confidence level with ratings. We give much more confidence to rating 5 for the movie "Scream" than the movie "Jeepers Creepers".

In this distribution-based algorithm, each rating is considered not as a single point, but as a probability distribution such as the one shown in Figure 2-2. The *x* value for the highest point of the curve is that rating, and the area under the curve is 1. Intuitively, if more nearest neighbors think an item is excellent, then it is more possible that the active user would like that item. We design the algorithm to simulate this situation. The prediction for each item is also a probability distribution created by "adding" probability distributions of nearest neighbors as illustrated in Figure 2-3.

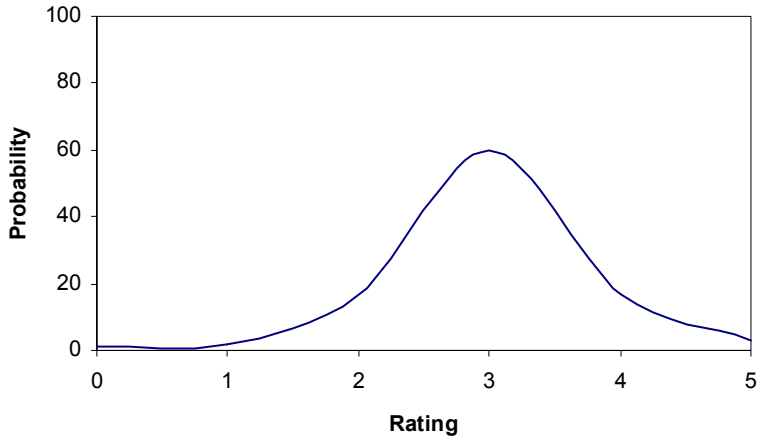

Figure 2-2 Probability distribution for rating 3

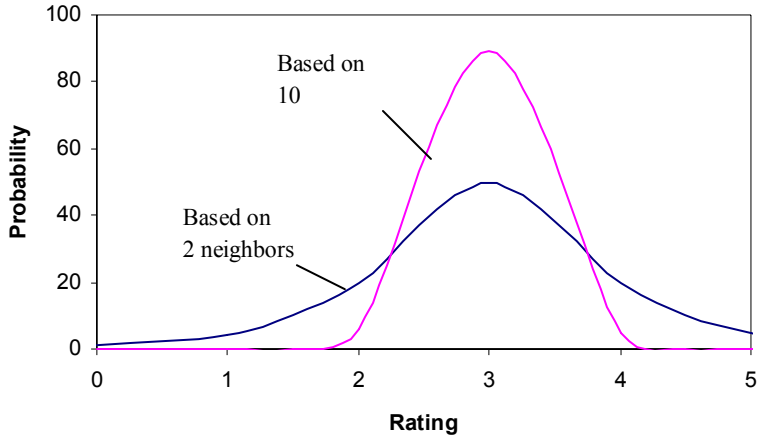

Figure 2-3 Different prediction probability distributions for rating 3

We design the algorithm as the following formula to calculate the possibility, $p(a, j)$, at which the active user, a, will give the rating j for the target item.

$$p(a, j) = \frac{u(j) + \sum_{i=1}^{n} f(i, j)}{n + 1} \tag{2-12}$$

Here, $j = 0, 1, \cdots, m$ represents the possible ratings. In our experiment, $m = 5$. $i = 1, 2, \cdots, n$ represents the nearest neighbors indexed from 1 to n.

$u(j)$ represents the default probability for the rating j, which can be treated in different ways. The simplest one is $1/(m+1)$ that means the user has the same probabilities to vote an item over all possible ratings. Alternatively, we can use the probability at which the target item is rated j over all users. Similarly, we can also use the probability at which the active user rates j over all items.

$f(i,j)$ represents the weighted possibility at which the neighbor *i* will give the rating *j* for the target item.

$$f(i,j) = w(a,i) \times p(i,j) + [1 - w(a,i)] \times u(j) \tag{2-13}$$

where $p(i,j)$ represents the possibility of *i*-th neighbor rating *j*. $w(a,i)$ is the weight representing the similarity measure between the active user and the *i*-th nearest neighbor. The Pearson correlation coefficient or cosine vector similarity could be used here. Considering its consistent performance, we apply the former one as the similarity measure.

$$w(a,i) = \frac{\sum_k (v_{a,k} - \bar{v}_a)(v_{i,k} - \bar{v}_i)}{\sqrt{\sum_k (v_{a,k} - \bar{v}_a)^2 \sum_k (v_{i,k} - \bar{v}_i)^2}} \tag{2-14}$$

where the summations over k include items that both user a and user i have rated in common.

Eq. (2-12), which is similar to Eq. (2-7), predicts the active user's rating behavior as the weighted sum of a set of the nearest neighbor's rating behaviors. Eq. (2-12) satisfies the constraint that the sum of all possibilities equals 1, which could be shown as follows.

Firstly,

$$\begin{aligned} \sum_{j=0}^{m} f(i,j) &= \sum_{j=0}^{m} \{w(a,i) \times p(i,j) + [1 - w(a,i)] \times u(j)\} \\ &= w(a,i) \times \sum_{j=0}^{m} p(i,j) + [1 - w(a,i)] \times \sum_{j=0}^{m} u(j) \\ &= w(a,i) \times 1 + [1 - w(a,i)] \times 1 = 1 \end{aligned} \tag{2-15}$$

where $\sum_{j=0}^{m} p(i,j)=1$ and $\sum_{j=0}^{m} u(j)=1$.

Secondely,

$$\sum_{j=0}^{m} p(a,j)=\sum_{j=0}^{m}\frac{u(j)+\sum_{i=1}^{n} f(i,j)}{n+1}=\frac{\sum_{j=0}^{m} u(j)+\sum_{j=0}^{m}\sum_{i=1}^{n} f(i,j)}{n+1}$$
$$=\frac{1+\sum_{i=1}^{n}\sum_{j=0}^{m} f(i,j)}{n+1}=\frac{1+\sum_{i=1}^{n} 1}{n+1}=\frac{1+n}{n+1}=1 \qquad (2\text{-}16)$$

Our approach differs from [Pennock et al., 2000] on the selection of the similarity measure. Our approach is based on the nearest neighbor method, while [Pennock et al., 2000] is based on the Bayesian clustering method.

To make a prediction, we need to find out a group of people with similar taste as the active user. We set the proper thresholds, the minimum number of co-rated items and least similarity, to select the neighbors.

The possibility, $p(i,j)$, at which the user i will rate j for a specific item could be converted as shown in Figure 2-4, where the different curves correspond to the relevant real rating values.

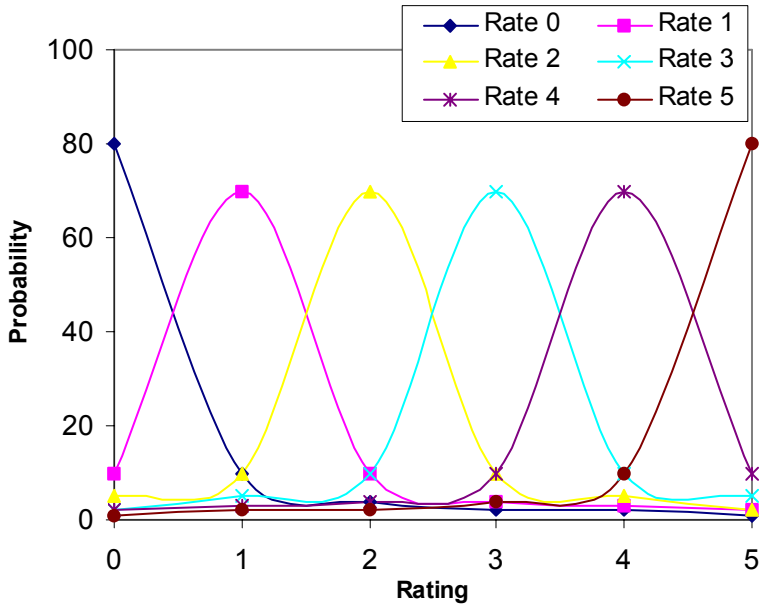

Figure 2-4 Different probability distributions for different ratings

2.3 EXPERIMENTAL IMPLEMENTATION

In this section, we describe the dataset used as well as the experimental methodology.

2.3.1 The EachMovie Dataset

We run experiments using the EachMovie dataset that was developed by Digital Equipment Corporation Research Center from 1995 through 1997. During that time, 72916 users entered a total of 2811983 numeric ratings for 1628 different movies, and each user rated 38 movies on average. User ratings were recorded on a

numeric six-point scale (0.0, 0.2, 0.4, 0.6, 0.8, 1.0) that maps linearly to zero-to-five star ratings. The data is publicly available and can be obtained from [7].

We only use a subset of the total data. We randomly choose 2000 users from the total data set, and each user at least rated 50 movies and on average rated 102 movies. Then these users are randomly divided into two sets, the training set and the test set, with 1000 users on each set. We restrict the number of users, not only because we consider the computation time and the certain protocols (i.e., Given-50), but also because we want to simulate the start phase for the recommender system in which the data does not contain so many users.

To analyze the performance of the proposed algorithm on the sparse data, we also randomly delete some ratings from the training set and reduce it to a smaller set, so that users in that set rate less number of movies than in the original training set.

2.3.2 Evaluation Metrics

Metrics measure how effectively the recommendations help users to choose high quality items from a huge item set. Although MAE (Mean Absolute Error) is a most widely used metrics to directly compare the predicted voting with the actual voting, we are more interested in whether the recommendations satisfy the users' need. For example, when a user asks for the excellent movies, we want to see if the recommended movies are truly very good. When a user asks for the bad movies to avoid them, we want to see if the returned movies are bad.

For this purpose, precision and recall are two useful metrics, and some researchers working on the collaborative filtering algorithms also use F1 metric, as defined in Eq. (2-19), which is equally weighted combination of precision and recall. Precision is the ratio of the number of relevant items retrieved to the total number of irrelevant and relevant items retrieved, while recall is the ratio of the

number of relevant items retrieved to the total number of relevant items in the dataset, as illustrated in Figure 2-5 [3].

$$F1 = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \tag{2-17}$$

In our experiment, we consider precision rather than recall, since we believe that the percentage of the recommended good items over all recommended items is more critical to the performance of the algorithms from the users' satisfactions' point of view – the bad recommendations may irritate users.

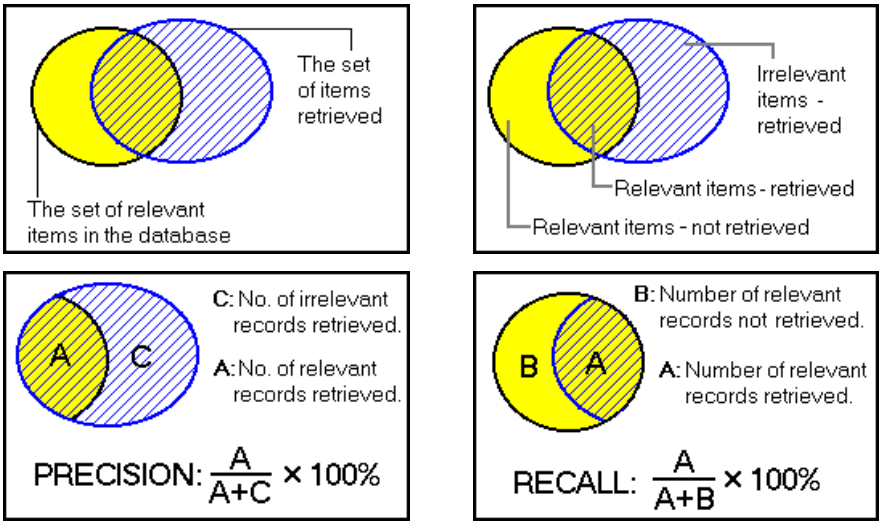

Figure 2-5 Precision and recall

2.3.3 Experimental Methodology

To compare with our proposed distribution-based algorithm, we implement the nearest neighbor method using the Pearson correlation as the similarity metric since it has reasonable accuracy and it is extensively used as a benchmark.

To evaluate the performance of the proposed algorithm on the dense and sparse data, we can select different number of available ratings for the active user according to different protocols and different number of available ratings for the neighbors by choosing different training sets.

We use five protocols similar to the method used by [Breese et al, 1998], Given-50, Given-40, Given-30, Given-20 and Given-10. We input the given number of ratings of an active user from the test set to the algorithm, and try to predict the remaining ratings of that user. Each protocol gives less information than the previous one. The accuracy using the nearest neighbor method decreases significantly when less information is available [Breese et al, 1998]. We use these protocols to see if the proposed algorithm can perform better on less data.

We also use five training sets, the original training set (on average 102 ratings per person), 1/2-training set (51 ratings per person), 1/4-training set (25 ratings per person), 1/6-training set (17 ratings per person) and 1/8-training set (12 ratings per person). We get 1/n-training set by randomly removing (n-1)/n of the ratings from the original training set.

Before testing the proposed algorithm, we define *excellent*, *very good*, *good*, *bad*, *terrible* and *extreme* movies in the EachMovie domain. Since we use the zero-to-five scale, we define *excellent* movies with rating five, *very good* movies with rating four or five, *good* movies with rating great than two, *bad* movies with rating less than two, *terrible* movies with rating zero and *extreme* movies with rating five or zero.

The experiments are designed to answer the following six questions:

1. Compare the ranked top-*N* result according to the possibility of rating 5 using the proposed algorithm with the result according to the predicted ratings in the decreasing order using the nearest neighbor method to answer the question "can the algorithm recommend *excellent* movies?"

2. Similar to 1, using the ranked result according to the sum of possibilities of ratings 4 and 5 to answer the question "can the algorithm recommend *very good* movies?"

3. Similar to 1, using the ranked result according to the sum of possibilities of ratings 3, 4 and 5 to answer the question "can the algorithm recommend *good* movies?"

4. Compare the ranked top-*N* result according to the sum of possibilities of ratings 0 and 1 using the proposed algorithm with the result according to the predicted ratings in the increasing order using the nearest neighbor method to answer the question "can the algorithm predict *bad* movies?"

5. Similar to 4, using the ranked result according to the possibility of rating 0 to answer the question "can the algorithm predict *terrible* movies?"

6. Compare the ranked top-*N* result according to the sum of possibilities of ratings 0 and 5 with the result according to the distances between the predicted ratings and the midpoint 2.5 in the decreasing order using the nearest neighbor method to answer the question "can the algorithm predict *extreme* movies?"

2.4 EXPERIMENTAL RESULTS

The proposed algorithm performs better in most cases than the nearest neighbor method using the Pearson correlation as a similarity metric, as shown in Table 2-1. The result is consistent in predicting *excellent* movies, *very good* movies, *good* movies, *bad* movies, *terrible* movies and *extreme* movies.

Table 2-1: Comparison results

Training Set \ Protocol	Given-10	Given-20	Given-30	Given-40	Given-50
Original	+	+	O	O	O
1/2	+	+	+	+	+
1/4	+	+	+	+	+
1/6	+	+	+	+	+
1/8	+	+	+	+	+

+ means significantly better than the nearest neighbor method
O means not significantly better than the nearest neighbor method

Some researches show that the accuracy using the nearest neighbor method will decrease when the active user provides less information [Breese et al, 1998], and this is also confirmed by our experiments using the different protocols. In Figure 2-6, the precision decreases 39.5% from 0.43 in Given-50 to 0.26 in Given-10 using the nearest neighbor method. However, the proposed algorithm performs almost the same under the different protocols, and the precision only changes from 0.43 to 0.44 in this case. This observations hold for all the training sets. This characteristic of the proposed algorithm is valuable for the practical recommender system. For an active user using a recommender system, there are "cost", providing his or her preference, and "benefit", receiving recommendations. If the "cost" is too expensive to get reasonable accuracy, some users may give up using the recommender system. The proposed algorithm gives users good predictions with a little "cost" so that the recommender system may attract more users.

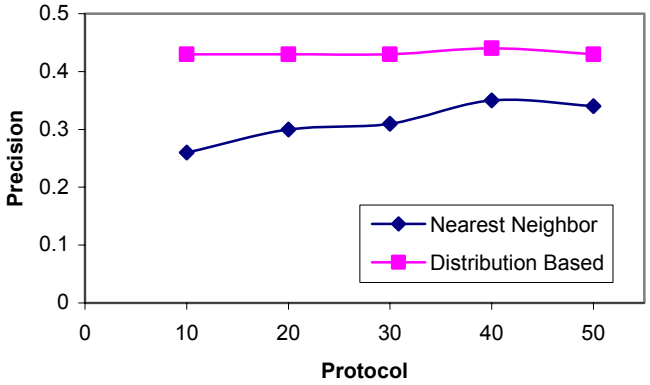

Figure 2-6 Precisions in 1/4-training set for Top-1

To evaluate the proposed algorithm on the sparse training set, we reduce the training set with 102 ratings per user on average gradually to the training set with 12 ratings per user. The proposed algorithm performs better than the nearest neighbor method, especially on the sparse training set. As shown in Figure 2-7, from the original dataset to 1/8-dataset, the precision decreases 54.2% from 0.48 to 0.22 using the nearest neighbor method, while the precision decreases only 21.2% from 0.52 to 0.41 using the proposed algorithm. The proposed distribution-based algorithm increases the accuracy much more significantly than the nearest neighbor method on the sparse data set, 1.86 (0.41/0.22) on the 1/8-dataset versus 1.08 (0.52/0.48) on the original dataset. This result is also valuable for the practical recommender system. The least data will be selected to maintain reasonable accuracy, since less data need less memory and less computing time.

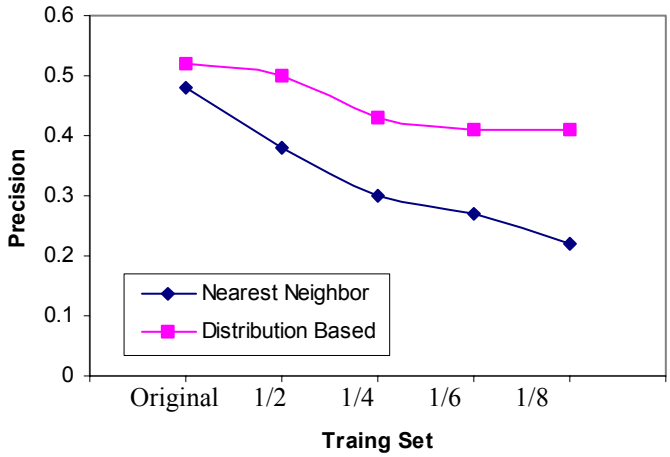

Figure 2-7 Precisions for Top-1 and Given-20 protocol

The distribution-based algorithm is better than the nearest neighbor method for all the ranked top-*N* results. The ratio of accuracy between these two algorithms reaches the highest in top-1, and the ratio is greater in the sparse dataset than the dense dataset. As shown in Figure 2-8, the ratio is 1.20 in top-1, 1.15 in top-5, 1.11 in top-10, 1.09 in top-15 and 1.1 in top-20, respectively, for the dense dataset (the original training set), while the ratio is 1.80 in top-1, 1.74 in top-5, 1.67 in top-10, 1.68 in top-15, and 1.67 in top-20, respectively, for the sparse dataset (1/8-training set).

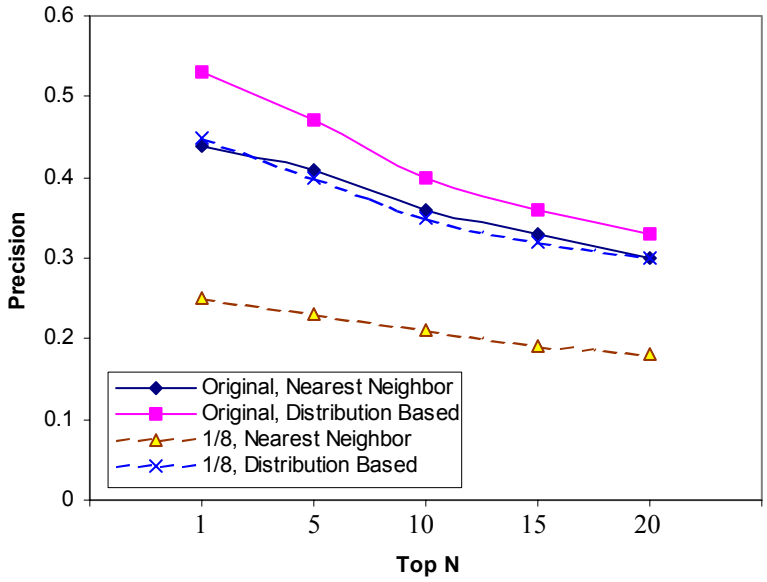

Figure 2-8 Precisions for Given 10

2.5 DISCUSSION AND CONCLUSION

In the proposed algorithm, we "add" the confidence of the rating according to the similarly between the active user and the nearest neighbors. Clearly, how to calculate the similarity has a strong effect on the accuracy. In our approach, we adopt the Pearson correlation coefficient, as it is considered as one of the most accurate similarity metrics. [Breese et al., 1998] As shown in Table 2-1, in most cases, the performance of the proposed algorithm is significantly better than the nearest neighbor method. However, for the dense training set (the original training set, 102 ratings per user) and the abundant information given by the active user (Given-30, Given-40 and Given-50), those two algorithms do not perform differently. One of the reasons may be that we do not take into account the co-rated number although we choose the nearest neighbor by a threshold of at least co-rated

2 movies. Intuitively, a neighbor with 3 co-rated movies with the correlation coefficient 0.80 is not better than a neighbor with 15 co-rated movies with the correlation coefficient 0.78. We test the following approaches based on the concept of significance weighting. [Herlocker et al., 1999]

1. Insert sqrt(co-rated number) into Eq. (2-12)

We introduce the square root of co-rated number into Eq. (2-12). To satisfy that the sum of possibilities for the active user to rate all values equals one, we change the formula as follows.

$$p(a,j) = \frac{u(j) + \sum_{i=1}^{n} \sqrt{n_{ai}} f(i,j)}{1 + \sum_{i=1}^{n} \sqrt{n_{ai}}} \tag{2-18}$$

where n_{ai} is the number of co-rated items between the active user and the *i*-th neighbor.

We run experiment using Eq. (2-18). The results show that this approach is not better than the original one. One possibility is that we add too much weight for the effect of co-rated number.

2. Use the Pearson correlation coefficient × sqrt(co-rated number)

According to the observation of a critical value table for the Pearson correlation coefficient, we find the product of the Pearson correlation coefficient and sqrt(co-rated number) is almost constant at the same significance level (*p* value). We consider the Pearson correlation coefficient × sqrt(co-rated number) as the

similarity metric and run experiments. The results show that this approach performs similar to the original one.

3. Use the Pearson correlation coefficient × sqrt(co-rated number / Given number)

$w(a,i)$ in the previous approach may be greater than one, so that the result of 1 - $w(a,i)$ may be negative, which is hard to explain intuitively. To overcome this problem, we substitute sqrt(co-rated number / Given number) for sqrt(co-rated number). The experimental results show that this approach also performs similar to the original one.

4. Use the Pearson correlation coefficient corresponding to co-rated 8 movies under the same *p* value

To avoid devaluing the weight of the neighbors with higher number of co-rated movies, we convert the original correlation coefficients to one with 8 co-rated movies under the same *p* value. For example, under $p = 0.05$, $p = 0.025$, $p = 0.01$ and $p = 0.005$, for $n = 8$ (*n* represents the number of co-rated movies), the correlation coefficients are 0.643, 0.738, 0.833 and 0.881, respectively; for $n = 10$, the coefficients are 0.564, 0.648, 0.745 and 0.794, respectively. We found that the formula of (1.0295 × the current correlation coefficient + 0.0657) can change the coefficients for $n = 10$ to those for $n = 8$. The experimental results show that this approach also performs similar to the original one.

We also test the following approaches to try to improve the accuracy.

For the default possibility, we use 1/6, the target item's rating probability over all users and the active user's rating probability over all items. There are no significant differences among them.

Inspecting the EachMovie data set, we found that the average rating value of each user has a large variance, and it is from 0.8 to 4.4. For example, Kent always gives higher ratings to the movies (*optimistic* users), while Chris always gives lower ratings to the movies (*pessimistic* users). That means the same ratings may have different interpretations for the different uses. We try to involve this effect into our formula. We horizontally translate the original distribution curve to the position that the average rating value equals 0. This approach does not significantly improve the accuracy over the original one.

All these unsuccessful efforts suggest a research issue to explore the possible upper accuracy boundary the prediction algorithms can reach, considering the intrinsic noise remained in the dataset. This is specifically important for the model-based algorithms to avoid over-fitting.

As a summary, in this section, we propose a new distribution-based algorithm that improves performance comparing with the nearest neighbor method using the Pearson correlation as a similarity metric, specially in the cases of the sparse training data, small n in Given-n and small N in Top-N. These features are useful for the actual recommender system.

3. COLLABORATIVE FILTERING WITH PRIVACY

Essentially, the data used in collaborative filtering is a matrix, whose rows represent the users, columns represent the items and each entry represents the relevant user's preference on the relevant item. The preferences can be obtained explicitly, directly asking users to give their opinions on some items, or can be obtained implicitly, for example, converting from users' web surfing behavior. While the data stored in this matrix is vital to make recommendations, it introduces the danger that the users' preferences may be maliciously used, and thus raises a serious privacy issue. For instance, a user's preferences may be used as the evidence against himself in the future. Also, those valuable data are considered as part of collectors' assets and may be routinely sold as such when collectors have suffered bankruptcy. [Canny, 2002] For instance, as stated in Amazon's privacy policy, "…in the unlikely event that Amazon.com, Inc., or substantially all of its assets are acquired, customer information will of course be one of the transferred assets." A survey released by Harris Interactive found that more Americans are concerned about loss of personal privacy (56 percent) than health care (54 percent), crime (53 percent) or taxes (52 percent) [4]. Therefore, protecting the users' privacy is crucial to further success of collaborative filtering.

In response to the growing concern about privacy, many solutions were proposed. However, as reviewed in the next section, none of them could reliably protect against violation of privacy. In the second half of this report, a novel approach is proposed, which could fundamentally eliminate any potential violations of privacy. This approach involves "blurring" each user's profile by adding some false preferences to the original data. As a result, only the general outline of each individual user will be stored and not any specific detail. To measure how well privacy is protected, the degree of obscurity and "evenly-adding" are introduced. Different approaches to add noise are presented, including adding noise randomly, adding noise according to user's behavior, adding noise using recommender

system, hybrid approach (i.e., adding noise 70% using recommender system plus 30% randomly) and adding noise using recommender system coupling with user's behavior.

This chapter is organized as follows. Firstly, we briefly review the existing solutions to protect privacy. Secondly, we describe the proposed "blurring" approach in detail. Then, we perform experiments to compare the prediction accuracy from different blurred datasets with that from the original data. The results show that although the original data is blurred the predictions are reasonably good and some approaches even increase the prediction accuracy.

3.1 LITERATURE REVIEW

In this section, we briefly review the existing solutions to protect privacy. Generally, there are two categories of solutions, namely, policy solutions and technology solutions.

Policy solutions include privacy policy, third-party oversight and P3P. Being encouraged business self-regulation by government, almost every web site posts its privacy policy. However, most of privacy policy is long, hard to understand and sometimes changes without notice. Third-party, such as BBBonline [2] and TRUSTe [8], monitor their licensee's compliance with its online privacy policy. P3P (Platform for Privacy Practices) [9] is a specification under development at the World Wide Web Consortium. It offers an easy way for web sites to communicate with users about their privacy policy in a standard machine-readable format. P3P-enabled browsers can automatically check privacy policy at web sites with a user's preference and inform result of comparison. However, all policy solutions have the so-called after-the-fact problem. It is too late for fixing the exposure of privacy and also sometimes the punishment is too weak. Additionally, even the information collectors themselves are willing to protect privacy, their abilities to securely store those valuable user data, mostly centralized, are doubtful.

Users are often using pseudonyms to surf the Internet, while it is not difficult to figure out the users' real name via their IP addresses. Anonymity, a technology solution, breaks the link between the requests to web site and the user's IP address. Anonymizing proxies, for example, Anonymizer [4], submit the requests for users so that only the proxy IP address is exposed to the requested web site. However, all users' activities are exposed to the anonymizing proxy. Crowds [6], named for the notion of "blending in to crowd" and developed by researchers at AT&T Labs, issues requests through a group of Crowds members and the true source of a request cannot be identified. But the reliability of message delivery is not guaranteed. Onion Routing [5], developed by researchers at Naval Research Lab, is based on the mix network, a collection of routers, which use techniques including encryption, buffering, reordering and equi-length to hide both the data being sent and who is talking to whom. But performance is heavily hurt. Anonymity will fail if the Web sites maintain the users' purchase records, which includes users' real name, address and even more sensitive information. More important, anonymity cannot link requests to individual directly, so it is hard to provide recommendations efficiently, especially based on the implicit ratings.

A similar solution to anonymity is proposed to protect privacy while enable personalization in [Arlein et al., 2000]. The abstraction of a persona is introduced, in which a user conducts the relevant web activity. The user can have many personae, such as "work", "entertainment" and "shopping". Via the persona abstraction, users control what information is grouped into a profile, and can selectively enable a Web site to read one or more of these profiles. To assist users in managing their personae, a persona server is introduced, which is separated from the profile database containing information about what a persona did while at different Web sites. Ideally, there are connections neither among personae nor between personae and users. This approach relies on the successful implementation of the persona server and the profile database, and the assumption that the Web sites follow their commitments, which have many uncertainties.

Recently, quite a few approaches are proposed to address the issue of privacy preservation in the context of data mining. Specifically, considering the following question: since the primary task in data mining is the development of models about aggregated data, can accurate models be developed without access to precise information in individual data records? An approach is presented to address this question by perturbing the original data and reconstructing distributions at an aggregate level in order to perform the mining. [Agrawal and Srikant, 2000] [Agrawal and Aggarwal, 2001] Formally, let us consider a set of n original data values $x_1, \ldots, x_n$, which are modeled as n independent values each drawn from the same data distribution as the random variable X. To create the perturbation, n independent values $y_1, \ldots, y_n$, each with the same distribution as the random variable Y, are generated. Thus, the perturbed values of the data are given by $z_1 = x_1 + y_1, \ldots, z_n = x_n + y_n$. In order to protect privacy, only the perturbed values are provided rather than the original data. Given these values and the (publicly known) density function $f_Y(y)$ for Y, the density function $f_X(x)$ for X is estimated.

At first glance, this approach closely resembles the mechanism adopted in this project. Both disturb the original user data before it is subject to the mining process to protect privacy. The former distorts each user's profile, i.e., by modifying the existing ratings, so only general patterns about a group of users can be seen, but not any individual user. Our approach, by contrast, blurs each user's profile by adding the new ratings, so although no any specific detail, general patterns about each individual user can be seen. Thus, the former approach is not suitable for making recommendations for users based on their profiles since the data are reconstructed at an aggregate level, not at an individual level.

To quantify the privacy provided by the different approaches, several privacy metrics are proposed with regard to the probability with which the user's distorted data can be reconstructed. The following metric is suggested in [Agrawal and Srikant, 2000]: if the original value can be estimated with c% confidence to lie in the interval $[\alpha_1, \alpha_2]$, then the interval width $(\alpha_2 - \alpha_1)$ defines the amount of privacy

at c% confidence level. As pointed out in [Agrawal and Aggarwal, 2001], this method does not take into account the fact that both the perturbed individual data and the aggregate distribution are available to the miner to make more accurate guesses about the possible values of the original data. A privacy metric is introduced in [Agrawal and Aggarwal, 2001] based on the concept of mutual information between the original and perturbed data. Given a random variable B, the conditional privacy loss of a random variable A is:

$$P(A|B) = 1 - 2^{-I(A;B)} \tag{3-1}$$

where $I(A; B)$ is the mutual information between the random variables A and B.

The following privacy metric is defined in [Rizvi and Haritsa, 2002], which is similar to our measurement proposed here.

$$P(p) = [1 - R(p)] \times 100 \tag{3-2}$$

where $R(p)$ is the total reconstruction probability, considering their Boolean matrix (i.e., 1 indicates a purchase and 0 indicates no purchase), given as:

$$R(P) = aR_1(p) + (1-a)R_0(p) \tag{3-3}$$

where $R_1(p)$ and $R_0(p)$ are the probabilities with which a '1' or a '0' can be reconstructed, respectively, and a is the weight denoting the preference which the privacy of 1's has over that of 0's.

Privacy-preserving data mining is also considered in the situation where the user data, specifically the matrix, are distributed across a number of sites, typically two parties, with each site only willing to share the mining results, but not the source data. In one case, the data is vertically partitioned (items split across sites),

i.e., each site hosts a disjoint subset of the matrix columns. [Vaidya and Clifton, 2002] In the other case, the data is horizontally partitioned (users split across sites), i.e., each site hosts a disjoint subset of the matrix rows. [Kantarcioglu and Clifton, 2002]

The idea expressed in the previous paragraph is extended to protect privacy in the context of collaborative filtering with multi-user. [Canny, 2002] As opposed to today's server-based collaborative filtering systems, a "user-owned and operated" principle is proposed: users have exclusive control and access to all data recorded about them. They should be able to control how and with whom the data will be shared, and be able to hide or restrict any part of the data. A protocol is suggested: users start with their own preference data, and then exchange various encrypted messages, back and forth within a community, hiding their peers' preference information. At the end of this process, each user has an unencrypted copy of the model of the community's preferences. Based on this aggregated model and her own ratings, the recommendations are derived for each individual user. Several issues emerge with this approach. First of all, although users are keen to maintain their profiles by themselves, their traces may be inevitably recorded in the Web sits they visited. Secondly, this approach requires a majority of honest users and totalers (who compute the totals). The hidden data should be valid and totalers' totals should be checked for accuracy. Thirdly, not all the collaborative filtering algorithms could be put in this scheme.

3.2 OBSCURITY APPROACH

In this section, we introduce the concept of obscuring profile to protect the recommender system users' privacy. To acquire the satisfied privacy while maintain the adequate accuracy, the degree of obscurity is proposed. The recommender system used here is based on the nearest neighbor method using the Pearson correlation as a similarity metric.

As mentioned earlier, the mechanism adopted here is to add false ratings on the users' un-rated items and thus to obscure the users' original profile. The following different approaches are used to add noise to the users' ratings.

RR - Adding random ratings to random movies. We randomly choose the movies the user did not rate, and then give them random ratings evenly distributed in the rating range.

RSD - Adding ratings to random movies according to the same distribution as the user's original ratings. We randomly choose the movies the user did not rate, and then give them ratings according to the relevant probabilities. For example, if in a user's original ratings 40% movies are rated as 3, it has 40% probability to give a un-rated movie a rating of 3.

RRS - Adding ratings to random movies according to recommender system's predictions. We consider the noise we added should somewhat similar to user's original ratings, so that it is difficult to differentiate between noise and real ratings. So we randomly choose the movies the user did not rate, and then give them ratings by recommender system.

The purpose of adding noise is to protect user's privacy. We should use some kind of metric to measure how well the privacy is protected. After adding noise, it should be hard to identify which ratings are users' real ratings and which are noise. This identification may be done using recommender system by setting some threshold. We make prediction for each movie and if the difference between the prediction and the rating is under threshold, we may think this rating is more likely the real one. Otherwise, this rating is more likely the added noise. These guesses may be wrong, which can be further divided into FP (False Positive) and FN (False Negative). FP means one rating is guessed as real while in fact it is added noise. On

the other hand, FN means one rating is guessed as added noise while in fact it is real.

Intuitively, more wrong guess, more protection provided to users' privacy. We can use the ratio of new approach wrong guess over the original wrong guess to measure how well users' privacy is protected. Since FP and FN may have different weight in different domain, the relevant weight should be considered when calculating the total wrong guess.

Thus, we introduce the degree of obscurity to measure the protection provided by the different approaches.

$$D = \frac{\lambda_{FP} FP_n + (1 - \lambda_{FP}) FN_n}{\lambda_{FP} FP_o + (1 - \lambda_{FP}) FN_o} \tag{3-4}$$

where D is the degree of obscurity, λ_{FP} represents the weight given to FP's over FN's, the subscripts n and o represent the noised and original data, respectively.

3.3 EXPERIMENTS AND RESULTS

Considering the computation efficiency, we use a subset of the EACHMOVIE dataset. The subset that we randomly choose consists of 98226 ratings of 2000 users on 1649 movies. Each user at least voted 11 movies (considering Given-10 protocol mentioned later) and on average 49 movies. To compare different approaches, we randomly divided the subset into the training set and the test set with 1000 users in each set.

We consider the training set as the data set we have already known, which may or may not contain noise. We pick one user each time from the test set as the active user (Step 1), who is supposed to be using this recommender system, and divide his ratings into two parts (Step 2). One part, which is added some noise (Step 3), is used as the input with the training set to the recommender system (Step 4) to make

the predictions for the other part (Step 5). Then comparisons are performed between the real ratings and the predicted ratings to measure the performance of different approaches (Step 6). The experiment procedure is sketched in Figure 3-1.

Three different protocols, proposed in [Breese, 1998], are employed to withhold part of ratings of an active user for comparisons, which reflect the different available amount of data.

All-but-one: each time pick one user from the test set, choose one original rating to predict, and add noise to the remaining ratings, then use the training set and these remaining obscured ratings to predict that chosen original rating.

Given-10: each time pick one user from the test set, choose ten original ratings and add noise to them, then use the training set and these obscured ratings to predict the remaining original ratings.

Given-5: similar to Given-10, just we choose five original ratings instead of 10.

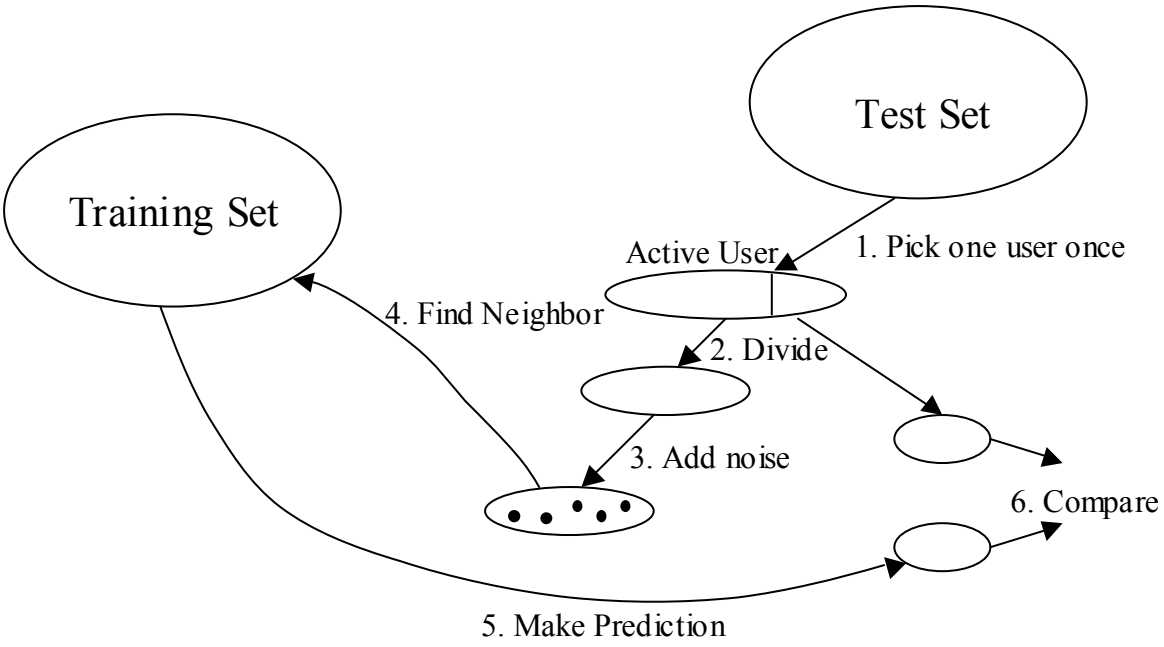

Figure 3-1 Experimental procedure

We adopt MAE (Mean Absolute Error) to measure the accuracy of different approaches, which is shown to draw the similar conclusions to other metrics [Herlocker, 2000]. The results are summarized in Table 3-1. Each case in this table is run for ten times and the value is the mean of the ten MAEs. MIX and RSSD will be discussed later.

A nonparametric method for constructing a hypothesis test *p*-value, namely the Wilcoxon rank method, as implemented in the S-PLUS function wilcox.test [S-Plus, 2001], is used to compare the relevant experimental results. The Wilcoxon rank method is better than the classical methods of statistical inference, such as Student's *t* methods, in terms of that the latter rely on the assumption that the data come from a normal distribution. The nonparametric methods work even when the actual distribution for the data is far from normal, that is, when the data do not have to have even a nearly normal distribution. The Wilcoxon rank method does rely on

the assumption that the observations within a sample are serially uncorrelated with one another. This assumption usually holds well in our experiments since the data are not collected in any specific times order.

The *p*-value, obtained from the Wilcoxon rank test, is the level of significance for which the observed test statistic value lies on the boundary between acceptance and rejection of the null hypothesis. For example, if the *p*-value obtained from the following S-PLUS function

wilcox.test(*vector1*, *vector2*, alternative = "greater")

is 0.03, the alternative hypothesis, the mean of the data, *vector1*, is greater than the mean of the data, *vector2*, is accepted at a significance level of 0.05, but not at a significance level of 0.01.

The following conclusions can be drawn from Table 3-1.

1. Under all situations the results from All-but-1 is better than Given-10 (*p*-value = 0.0017), and in turn than Given-5 (*p*-value = 0.0016), since more information about the active user provided to recommender system, more accurate predictions we can obtain. This result shows the same trend as those reported in [Breese, et al., 1998].

Table 3-1: Experimental results

Approach	Noise [%]	Protocol		
		All-but-1	Given-10	Given-5
None	0	0.975	1.130	1.243
RR	25	0.982	1.122	1.241
	50	0.974	1.134	1.237
	75	0.963	1.136	1.254
	100	0.994	1.146	1.256

RSD	25	0.989	1.119	1.227
	50	0.978	1.130	1.240
	75	0.964	1.127	1.237
	100	0.983	1.132	1.244
RRS	25	0.983	1.120	1.226
	50	0.971	1.115	1.222
	75	0.956	1.104	1.214
	100	0.971	1.104	1.198
MIX	25	0.961	1.118	1.233
	50	0.983	1.112	1.219
	75	0.982	1.107	1.219
	100	0.986	1.104	1.220
RSSD	25	0.975	1.120	1.229
	50	0.979	1.119	1.237
	75	0.976	1.116	1.235
	100	0.968	1.114	1.230

2. The nearest neighbor method we used is robust to the noise. MAE has a reasonable accuracy even adding 100% noise to the original data, comparing with the results reported in [Breese, et al., 1998].

3. Adding noise using RR and RSD will decrease prediction accuracy a little and the result from RSD is slightly better than RR if we add the same percentage noise, especially when 75% and 100% noise is added (p-value = 0.0625). The reason we believe is that added noise in RSD may have more probabilities to coincidentally reflect the active user's preference. With the increase of the added noise percentage, the accuracy of RR decreases faster than RSD.

4. Adding noise using RRS will even increase prediction accuracy in the cases of All-but-1 with adding 300% noise (p-value = 0.0185) and Given-10 and Given-5 (p-value < 0.05). This shown that adding noise by RRS enhances the active user's main interests to better select neighbors and make predictions. With the increase of the added noise percentage, the accuracy of RRS will also constantly increase up to

100% noise added. One reason of increasing accuracy is illustrated using the following example. Suppose users Tom and Mark have very similar taste. However, Tom only rated movies A and B, Mark only rated movies C and D, so we can not select Mark as a neighbor for Tom. If we add noise to Tom according to recommender system's predictions, now Tom has ratings on movies C and D, which are similar to Mark's ratings on C and D. Then we can select Mark as a neighbor for Tom. Intuitively, this may increase the prediction accuracy since more implicit nearest neighbors are selected.

5. Adding noise by RRS increase the accuracy of prediction especially in the sparse data case. As seen from Table 3-1, added noise in Given-5 and Given-10 protocols increase accuracy more significant than All-but-1. In the case of All-but-1, only when larger noises (300%) are added to the original data, adding noise by RRS can increase the accuracy significantly. In the cases of Given-10 and Given-5, even 25% noises can increase the accuracy significantly.

Figure 3-2 and Figure 3-3 show the wrong guess percentage and degree of obscurity for the different approaches, which reflect the protection degree. The horizontal axis in Figure 3-2 and Figure 3-3 is threshold, which is used to determine the data is the real rating or the added noise.

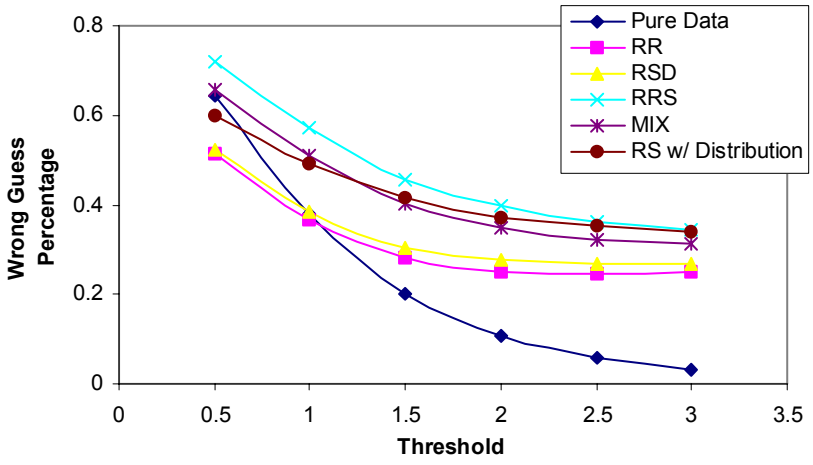

Figure 3-2 Wrong guess percentage in the case of adding 50% noise

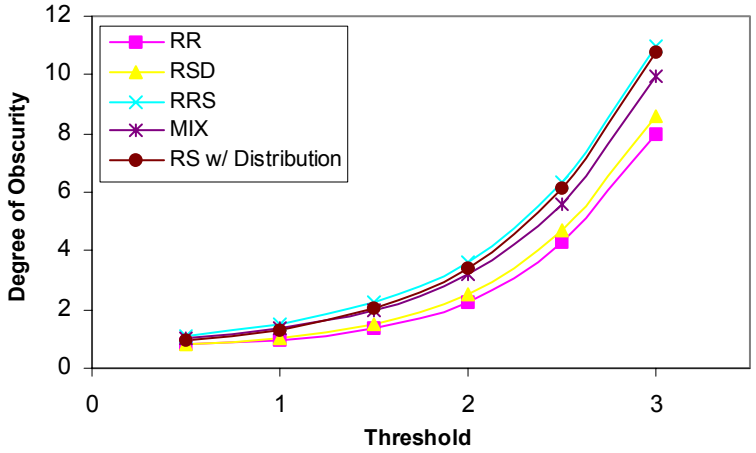

Figure 3-3 Degree of obscurity in the case of adding 50% noise

The following conclusions can be drawn from Figure 3-2 and Figure 3-3.

1. RRS obtains best results in terms of degree of obscurity. RSD is a little better than RR.

2. After the threshold is greater than 1, all approaches can protect the users' privacy since all the degrees of obscurity are greater than 1.

RRS can protect the active users' main preference, however, it may not protect the users' original unusually aberrant or deviant data values, which is shown in Figure 3-4. Almost all noise is added below the threshold about 1. Thus, if we know how the noise is added, we may guess all ratings which are beyond this threshold are real ratings. To take this into account, noise should be added more evenly to the main and strange preference. We introduce the degree of "evenly-adding", which is the inverse of the standard derivation of the percentage of noise in each threshold range.

To overcome this problem, two more approaches are proposed.

MIX - We randomly choose the movies the user did not rate, and then rate 30% of them by RR and 70% of them by RRS.

RRSD - We randomly choose the movies the user did not rate, and then instead of rating them directly by the recommender system's prediction, we rate them by adding a distance to this prediction according to the original distance distribution shown in Figure 3-4.

MAEs from these two approaches are shown in Table 3-1. We can see that they almost have the same accurate as RRS. The wrong guess percentage and degree of obscurity are shown in Figure 3-2 and Figure 3-3. Their performance are very close to RRS, while they increase the degree of evenly-adding to better protect the users' privacy, which is shown in Figure 3-5 and Figure 3-6.

The calculated degrees of evenly adding are 2.38 for RRS, 3.69 for MIX and 8.00 for RRSD, respectively. Obviously, RRSD adds noise more evenly.

To evaluate an approach, both degree of obscurity and evenly adding should be considered.

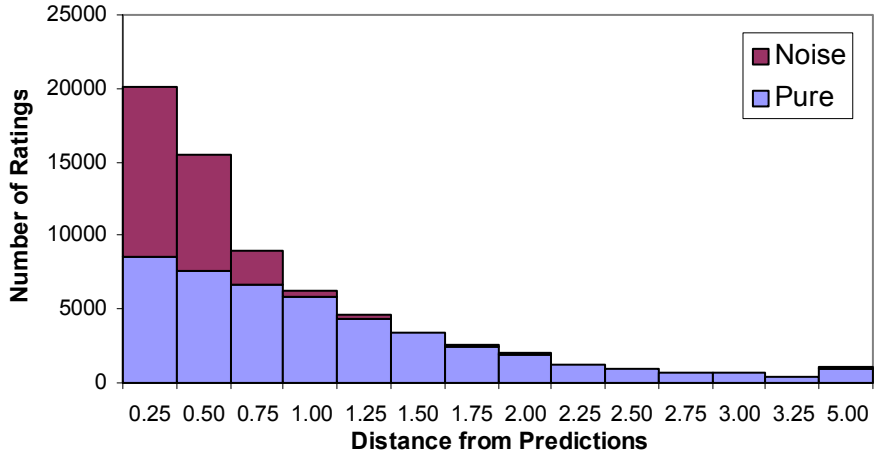

Figure 3-4 RRS with 50% noise

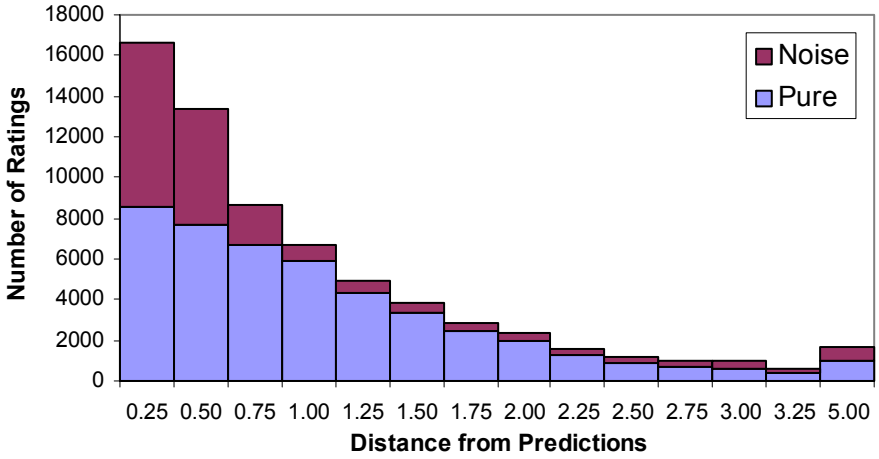

Figure 3-5 MIX with 50% noise

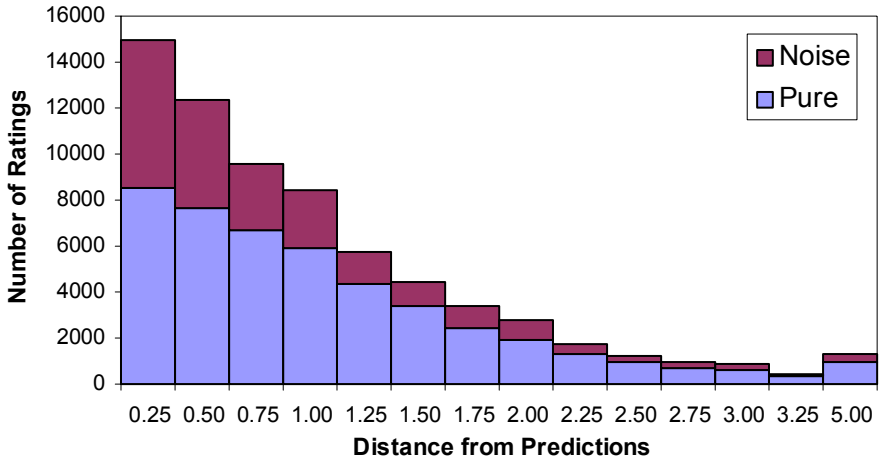

Figure 3-6 RRSD with 50% noise

Based on the above observations, the following conclusions can be drawn.

1. The proposed approaches can protect the users' privacy, and at the same time, they can maintain the satisfied accuracy. RRS may even increase the prediction accuracy.

2. To measure how well the approaches protect the users' privacy, degree of obscurity and evenly-adding are introduced, where, the former is to protect the users' main interests while the latter is to protect the users' strange interests.

3. MIX and RRSD increase both of these two parameters, and have almost the same performance (the *p*-value for their MAE comparison is 0.1575). Thus, MIX and RRSD are suggested to be used in the real world to protect uses' privacy.

4. CONCLUSIONS AND POSSIBLE FUTURE WORK

Collaborative filtering has seen considerable success in the areas regarding information overload and e-commerce, while the current developed systems are flawed in several respects. Two approaches, the distribution-based algorithm and the blurring profile solution, are proposed to address several outstanding issues.

The main findings of this research include:

- The proposed distribution-based algorithm, like the traditional nearest-neighbor method, is easy to understand and easy to implement, while it can provide a confidence level for each predicted rating.
- Compare to the nearest-neighbor method using the Pearson correlation coefficient as a similarity metric, the distribution-based algorithm performs significantly better, especially for the sparse dataset, where less information about the active user and the neighbors are available.
- A technology-based solution, blurring each user's profile by inserting new ratings that did not actually exist, is proposed to protect users' privacy. To quantify privacy provided by different approaches, the degrees of obscurity and evenly-adding are introduced, where the former is to protect the users' main interests while the latter is to protect the users' strange interests.
- The proposed approaches can protect the users' privacy, at the same time, they can maintain the satisfied accuracy, and in some cases, they can even increase the prediction accuracy. This mechanism may be used to fill the missing data, which is required for the most model-based algorithms.

With the knowledge and experience we gained from this study, we felt that there are certain improvements and modifications for further work. Those future efforts are proposed below:

- The experiments performed here are based on the explicit rating dataset, specifically the EachMovie dataset. It is suggested to carry out more experiments on the implicit rating dataset.
- The blurring profile solution proposed to protect privacy is tested with the nearest neighbor method. More work is suggested to test its performance with other algorithms, i.e., the model-based algorithms.
- The noise intrinsically existed in the original dataset is an interesting research issue, which may be considered as the upper accuracy boundary the prediction algorithms can reach. This is specifically important for the model-based algorithms to avoid over-fitting.

BIBLIOGRAPHY

1. http://www.anonymizer.com/
2. http://www.bbbonline.org/
3. http://www.hsl.creighton.edu/hsl/Searching/Recall-Precision.html
4. http://www.nclnet.org/pressessentials.htm
5. http://onion-router.nrl.navy.mil/
6. http://www.research.att.com/projects/crowds/
7. http://research.compaq.com/SRC/eachmovie/
8. http://www.truste.org/
9. http://www.w3.org/P3P/
10. [Agrawal and Aggarwal, 2001] Agrawal, D. and Aggarwal, C. C. (2001) On the Design and Quantification of Privacy Preserving Data Mining Algorithms. Proceedings of Twentieth ACM Symposium on Principles of Database Systems, Santa Barbara, CA.
11. [Agrawal and Srikant, 2000] Agrawal, R. and Srikant R. (2000) Privacy-Preserving Data Mining. Proceedings of ACM SIGMOD International Conference on Managements of Data, Dallas, TX.
12. [Arlein et al., 2000] Arlein, R. M., Jai, B., Jakobsson, M., Monrose, F. and Reiter, M. K. (2000) Privacy-Preserving Global Customization. Proceedings of the Second ACM conference on Electronic commerce, Minneapolis, MN.
13. [Breese et al., 1998] Breese, J., Heckerman, D. and Kadie, C. (1998) Empirical Analysis of Predictive Algorithms for Collaborative Filtering. Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Madison, WI.
14. [Canny, 2002] Canny, J. (2002) Collaborative Filtering with Privacy via Factor Analysis. ACM SIGIR, Tampere, Finland.

15. [Goldberg et al., 1992] 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).

16. [Goldberg et al., 2001] Goldberg, K., Roeder, T., Gupta, D. and Perkins, C. (2001) Eigentaste: A Constant Time Collaborative Filtering Algorithm. Information Retrieval 4, 2

17. [Greening, 2000] Greening, D. R. (2000) Building Consumer Trust with Accurate Product Recommendations: A White Paper on Macromedia's LikeMinds Personalization Technology. Boston: Macromedia.

18. [Heckerman et al., 2000] Heckerman, D., Chickering, D., Meek, C., Rounthwaite, R. and Kadie, C. Dependency Networks for Inference, Collaborative Filtering, and Data Visualization. Journal of Machine Learning Research, 1.

19. [Herlocker, 2000] Herlocker, J. Understanding and Improving Automated Collaborative Filtering Systems. Ph.D Dissertation, University of Minnesota, 2000.

20. [Herlocker et al., 1999] Herlocker, J., Konstan, J., Borchers, A. and Riedl, J. (1999) An Algorithmic Framework for Performing Collaborative Filtering. Proceedings of the 1999 SIGIR Conference on Research and Development in Information Retrieval, Berkeley, CA.

21. [Kantarcioglu and Clifton, 2002] Kantarcioglu, M. and Clifton, C. (2002) Privacy-preserving Distributed Mining of Association Rules on Horizontally Partitioned Data. Proceedings of ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, Madison, WI.

22. [Lyman and Varian, 2000] Lyman, P. and Varian, H. R. (2000) How Big Is the Information Explosion? iMP Magazine: http://www.cisp.org/imp/november_2000/11_00lyman.htm.

23. [Oracle, 2001] Oracle. (2001) Getting Started with Oracle9*i* Personalization.

24. [Parkes, 2001] Parkes, C. (2001) The Power of Personalization. Enterprise Systems.

25. [Pennock et al., 2000] Pennock, D. M., Horvitz, E., Lawrence, S. and Giles, C. L. (2000) Collaborative Filtering by Personality Diagnosis: A Hybrid Memory-

and Model-Based Approach. Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, San Francisco, CA.

26. [Rizvi and Haritsa, 2002] Rizvi, S., J. and Haritsa, J. R. (2002) Maintaining Data Privacy in Association Rule Mining. Proceedings of the Twenty-Eighth VLDB Conference, Hong Kong, China.

27. [Sarwar et al., 2000] Sarwar, B., Karypis, G., Konstan, J. and Riedl, J. (2000) Application of Dimensionality Reduction in Recommender Systems - A Case Study. Proceedings of the WebKDD Workshop at the ACM SIGKKD, Boston, MA.

28. [S-Plus, 2001] Insightful Corporation. (2001) S-Plus 6 for Windows: Guide to Statistics. Seattle, WA.

29. [Vaidya and Clifton, 2002] Vaidya, J. and Clifton, C. (2002) Privacy Preserving Association Rule Mining in Vertically Partitioned Data. Proceedings of the Eighth ACM SIKGDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Canada.

30. [Waddington, 1996] Waddington, P. (1996) Dying For Information?

31. [Yu et al., 2001] Yu, K., Wen, Z., Xu, X. and Ester, M. (2001) Feature Weighting and Instance Selection for Collaborative Filtering. Proceedings of the Second International Workshop on Management of Information on the Web - Web Data and Text Mining, Munich, Germany.