Nathan N. Liu and Qiang Yang. In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '08). ACM, New York, NY, USA, 83-90. 2008.

EigenRank: A Ranking-Oriented Approach to Collaborative Filtering

Nathan N. Liu
Department of Computer Science and Engineering
Hong Kong University of Science and Technology, Hong Kong, China
nliu@cse.ust.hk

Qiang Yang
Department of Computer Science and Engineering
Hong Kong University of Science and Technology, Hong Kong, China
qyang@cse.ust.hk

ABSTRACT

A recommender system must be able to suggest items that are likely to be preferred by the user. In most systems, the degree of preference is represented by a rating score. Given a database of users' past ratings on a set of items, traditional collaborative filtering algorithms are based on predicting the potential ratings that a user would assign to the unrated items so that they can be ranked by the predicted ratings to produce a list of recommended items. In this paper, we propose a collaborative filtering approach that addresses the item ranking problem directly by modeling user preferences derived from the ratings. We measure the similarity between users based on the correlation between their rankings of the items rather than the rating values and propose new collaborative filtering algorithms for ranking items based on the preferences of similar users. Experimental results on real world movie rating data sets show that the proposed approach outperforms traditional collaborative filtering algorithms significantly on the NDCG measure for evaluating ranked results.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

Ranking, Collaborative Filtering, Random Walk

1. INTRODUCTION

With the information available to us growing far more rapidly than our ability to process it, technologies to help people sift through huge amount of information efficiently is becoming increasingly important in order to overcome the resulted information overload problem. Recommender system is one such promising technology that aims to generate item recommendations from a huge collection of items based on users' preferences. Broadly speaking, existing technologies used for recommender systems fall in either of the following two categories: *content-based filtering* versus *collaborative filtering*. Content-based filtering approach analyzes the content information associated with the items and users such as product descriptions, user profiles etc. in order to represent users and items using a set of features. To recommend new items to a user, content-based filters match their representations to those items known to be of interest to the user. In contrast, the collaborative filtering(CF) approach does not require any content information about the items, it works by collecting ratings on the items by a large number of users and make recommendations to a user based on the preference patterns of other users. The CF approach is based on the assumption that a user would usually be interested in those items preferred by other users with similar interests. Besides avoiding the need for collecting extensive information about items or users, CF requires no domain knowledge and can be easily adopted in different recommender systems.

Collaborative filtering is usually adopted in two classes of application scenarios[2]. In the first class, a user is presented with one individual item at a time along with a predicted rating indicating the user's potential interest in the item. An example of this category is GroupLens[17], a collaborative filtering system for Usenet news. The second class of applications produce an ordered list of Top-N recommended items where the highest ranked items are predicted to be most preferred by the user. The user is expected to examine the items in the list starting from the top positions. Many existing E-Commerce Websites fall in this category. In this paper, we focus on improving collaborative filtering techniques for the second class of applications.

The computation of the Top-N item list for making recommendations is essentially a ranking problem. To produce a ranking of the items, most collaborative filtering algorithms adopt a rating-oriented approach which first predicts the potential ratings a target user would assign to the items and then rank the items according to the predicted ratings. However, higher accuracy in rating prediction does not necessarily lead to better ranking effectiveness as illustrated in the following simple example. Suppose we have two items i and j for which the true ratings are known to be 3 and

4 respectively and two different methods have predicted the ratings on i and j to be $\{2, 5\}$ and $\{4, 3\}$ respectively. In terms of rating prediction accuracy as measured by the absolute deviation from the true rating, there is no difference between the two sets of predictions. However, using the predictions $\{4, 3\}$, item i and j will be incorrectly ordered while the predictions $\{2, 5\}$ ensures the correct order. The problem with rating-oriented CF approaches is that the focus has been placed on approximating the ratings rather than the rankings, which is a more important goal for recommender systems. Moreover, most existing methods predict the ratings for each individual items independently without considering the user's preferences regarding pairs of items.

In this paper, we propose a ranking-oriented approach to collaborative filtering that directly addresses the item-ranking problem without going through the inter-meditate step of rating prediction. The main contributions of this paper is first to describe a similarity measure for evaluating the consistency between two user's rankings of a set of items that can be used to determine a set of users that share similar preferences to the target users. We then present two methods for producing item rankings based on the preferences of a set of similar users: one is a greedy algorithm that searches through the possible rankings for the optimal one and the other is a novel random walk model defined using a user's preference information. Both methods aim to effectively combine the partial and incomplete item rankings derived from the ratings of a set of similar users in order to rank the items in a way that is most consistent with all the known information about user preferences, which is why we name our approach "EigenRank".

This paper is organized as follows: in section 2, we briefly review some related works. Section 3 describes in more detail several similarity measures and models in traditional rating oriented CF approach. We then present the similarity measures and models for the proposed ranking-oriented CF approach in section 4. The experimental results and analysis are shown in section 5. Finally, we conclude the paper in section 6.

2. RELATED WORKS

There are two types of common approaches to collaborative filtering. One is the neighborhood-based approach and the other is the model-based approach.

2.1 Neighborhood-based Approaches

The most common form of neighborhood-based approach is the user-based model, which estimate the unknown ratings of a target user based on the ratings by a set of neighboring users that tend to rate similarly to the target user. A crucial component of the user-based model is the user-user similarity $s_{u,v}$ that is used to select the set of neighbors. Popular choices for $s_{u,v}$ include the Pearson Correlation Coefficient(PCC)[22, 11]and the vector similarity(VS)[2].

One difficulty in measuring the user-user similarity is that the raw ratings may contain biases caused by the different rating behaviors of different users. For example, some users may tend to give high ratings. To correct such biases, different methods have been proposed to normalize or center the data prior to measuring user similarities. [22, 2] showed that by correcting for user-specific means the prediction quality could be improved. Later, Jin et al. proposed a technique for normalizing the user ratings based on the halfway cumulative distribution[15].

Another difficulty in user-based models arises from the fact that the known user-item ratings data is typically highly sparse, which makes it very hard to find highly similar neighbors for making accurate predictions. To alleviate such sparsity problem, different techniques have been proposed to fill in some of the unknown ratings in the matrix such as dimensionality reduction[8] and data-smoothing methods[25, 19].

An alternative form of the neighborhood-based approach is the item-based model[24, 18]. Here the item-item similarity is used to select a set of neighboring items that have been rated by the target user and the ratings on the unrated items are predicted based on his ratings on the neighboring items. Since the number of items is usually much less than the number of users in most applications, item-item similarities are less sensitive to the data sparsity problem. Sarwar et al.[24] recommended using the adjusted cosine similarity to compute the item-item similarity and found that the item-based model could obtain higher accuracy than the user-based model, while allowing more efficient computations.

2.2 Model-based Approaches

In contrast to the neighborhood-based approach, the model-based approach to CF use the observed user-item ratings to train a compact model that explains the given data so that ratings could be predicted via the model instead of directly manipulating the original rating database as the neighborhood-based approach does. Algorithms in this category include clustering methods[25], aspect models[12] and Bayesian networks[21].

2.3 Learning to Rank

Learning to rank has been attracting broad attention in the machine learning community due to its importance in a wide variety of applications such as information retrieval, collaborative filtering, etc. Most of the proposed methods are dedicated to ranking items represented in some feature space as is the setting for content-based filtering. Given a set of ordered pairs of instances as training data, the different methods either try to learn an item scoring function[4, 16] or learn a classifier for classifying item pairs into two types of relations(correctly ordered vs. incorrectly ordered)[5, 6]. Different machine learning models including SVM, Boosting and Neural Network have been used for learning such ranking functions, which led to methods such as Ranking SVM[16], RankNet[4] and RankBoost[6].

3. RATING ORIENTED COLLABORATIVE FILTERING

In this section, we first describe several rating-based similarity measures that have been commonly used in neighborhood-based CF approaches for finding similar users[22, 2] and similar items[24, 18]. We then discuss two models for rating prediction in neighborhood-based CF, namely user-based and item-based models. The notational framework is defined as follows. Suppose we are given a set of m users $U = \{u_1, u_2, ..., u_m\}$ and a set of n items $I = \{i_1, i_2, ..., i_n\}$. The user's ratings on the items can be represented by an $m \times n$ matrix R where each entry $r_{u,i}$ represents user u's rating on item i and $r_{u,i} = 0$ if u has not rated i. The set of users who have rated item i is denoted by U_i and the set of items that have been rated by user u is denoted by I_u.

3.1 Pearson Correlation Coefficient

The Pearson Correlation Coefficient(PCC) measures the similarity between two users based their normalized ratings on the set of items they rated in common:

$$s_{u,v} = \frac{\sum\limits_{i\in I_u\cap I_v} (r_{u,i} - \overline{r}_u)(r_{v,i} - \overline{r}_v)}{\left[\sum\limits_{i\in I_u\cap I_v} (r_{u,i} - \overline{r}_u)^2 \sum\limits_{i\in I_u\cap I_v} (r_{v,i} - \overline{r}_v)^2\right]^{1/2}} \tag{1}$$

3.2 Vector Similarity

Another way of measuring user-user similarity is to view each user as a vector in a high dimensional vector space based on his ratings so that the cosine of the angle between the two corresponding vectors can be used to measure their similarities:

$$s_{u,v} = \frac{\sum\limits_{i\in I_u\cap I_v} r_{u,i}\cdot r_{v,i}}{\left[\sum\limits_{i\in I_u\cap I_v} r_{u,i}^2 \sum\limits_{i\in I_u\cap I_v} r_{v,i}^2\right]^{1/2}} \tag{2}$$

For measuring item-item similarity in item-based models, the adjusted cosine similarity has been shown to be most effective [24]:

$$s_{i,j} = \frac{\sum\limits_{u\in U_i\cap U_j} (r_{u,i} - \overline{r}_u)(r_{u,j} - \overline{r}_u)}{\left[\sum\limits_{u\in U_i\cap U_j} (r_{u,i} - \overline{r}_u)^2 \sum\limits_{u\in U_i\cap U_j} (r_{u,j} - \overline{r}_u)^2\right]^{1/2}} \tag{3}$$

where each user's rating on an item is adjusted by his mean rating.

3.3 Rating Prediction

In user-based collaborative filtering, the unknown ratings $\widehat{r}_{u,i}$'s are predicted by selecting a set of k most similar users N_u based on the user-user similarities $s_{u,v}$ and compute the weighted average of the ratings on i assigned by users in N_u:

$$\widehat{r}_{u,i} = \overline{r}_u + \frac{\sum\limits_{v\in N_u\wedge U_i} s_{u,v}(r_{v,i} - \overline{r}_v)}{\sum\limits_{v\in N_u\wedge U_i} s_{u,v}} \tag{4}$$

Following the same idea, in the item-based model, $\widehat{r}_{u,i}$'s are predicted based on u's ratings on items in N_i, the set of k items most similar to i determined using the item-item similarities $s_{i,j}$:

$$\widehat{r}_{u,i} = \frac{\sum\limits_{j\in N_i\wedge I_u} s_{i,j}\cdot r_{v,i}}{\sum\limits_{j\in N_i\wedge I_u} s_{i,j}} \tag{5}$$

4. RANKING-ORIENTED COLLABORATIVE FILTERING

Traditional rating-oriented collaborative filtering focuses on predicting a user's potential ratings on unrated items by utilizing the known ratings associated with similar users or similar items. In this section, we present a ranking-oriented collaborative filtering approach that aims at producing an item ranking for the target user. We first describe a user-user similarity measure that is based on two users' preferences over the items and then present two methods for ranking items based on the preferences of the set of neighbors of the target user.

4.1 Kendall Rank Correlation Coefficient

Both PCC and VS described in Section 3 are rating-based similarity measures in the sense that they are calculated by comparing ratings values assigned to the items by different users. In the ranking-oriented approach, the similarity between users is determined by their preferences over the items, which is reflected by their ranking of the items. Suppose we have a set of three items, to which two users have assigned ratings of {2,3,4} and {3,4,5} respectively. The rating values on the same items by the two users are clearly different, nevertheless their preferences are very close as the items are ordered in the way based the two user's ratings. The Kendall rank correlation coefficient[20] is a similarity measure between two rankings of the same set of objects:

$$s_{u,v} = 1 - \frac{4\times \sum\limits_{i,j\in I_u\cap I_v} I^{-}\big((r_{u,i} - r_{u,j})(r_{v,i} - r_{v,j})\big)}{|I_u\cap I_v|\cdot(|I_u\cap I_v| - 1)} \tag{6}$$

where $I^{-}(x)$ is an indicator function defined as:

$$I^{-}(x) = \begin{cases} 1 & \text{if } x < 0 \\ 0 & \text{otherwise} \end{cases}$$

The value of the coefficient is negatively correlated with the number of disconcordant pairs, where a pair of items i and j is disconcordant if i is ranked higher than j in one ranking but lower in the other.

4.2 Preference Functions

Since our goal is to produce a ranking of the items for a user rather than predicting the rating values, we focus on modeling a user's preference function of the form $\Psi : I\times I \rightarrow \mathbb{R}$, where $\Psi(i,j) > 0$ means that item i is more preferable to j for user u and vice versa. The magnitude of this preference function $|\Psi(i,j)|$ indicates the strength of preference and a value of zero means that there is no preference between the two items. Following [6], we assume that $\Psi(i,i) = 0$ for all $i \in I$ and that Ψ is anti-symmetric, i.e. $\Psi(i,j) = -\Psi(j,i)$ for all $i,j \in I$. Note that, however, we do not require Ψ to be transitive, i.e. $\Psi(i,j) > 0 \wedge \Psi(j,k) > 0$ does not imply $\Psi(i,k) > 0$.

In content-based filtering, the preference function Ψ is often realized by a binary classifier that categorizes each pair of items into two categories(correctly ranked and incorrectly ranked) based on their content features. Various machine learning approaches including ensembles[5, 6], support vector machines[16] and neural networks[4] have been developed for learning such a binary classifier to model the preference function. However, in the collaborative filtering setting, such approaches could not be applied due to the lack of features for describing the items.

Given a user's ratings on a set of items, we can derive his preference over the items by comparing his ratings on pairs of rated items. Suppose user u's rating on item i and j are 5 and 3 respectively, this clearly indicates that he prefer i to j and should serve as evidence for $\Psi(i,j) > 0$ and $\Psi(j,i) < 0$. In ranking-oriented collaborative filtering, the key challenge is to obtain preference information regarding pairs of items that have not both been explicitly rated by the target user. Following the same idea of neighborhood-based collaborative filtering, we resort to a set of users with similar preferences to the target user, referred to as the neighborhood N_u. The basic idea is that the more often the users in N_u assign i a higher rating than j, the stronger the evidence for $\Psi(i,j) > 0$ and $\Psi(j,i) < 0$. This leads to the following formula for estimating the values of the preference function $\Psi(i,j)$:

$$\Psi(i,j) = \frac{\sum_{v \in N_u^{i,j}} s_{u,v} \cdot (r_{v,i} - r_{v,j})}{\sum_{v \in N_u^{i,j}} s_{u,v}}$$

where the summation is over $N_u^{i,j}$, the set of neighbors of u who have rated both item i and j.

Given a preference function Ψ, which assigns a score to every pair of items $i,j \in I$, we want to choose a ranking of items in I that agrees with the pairwise preferences defined by Ψ as much as possible. Let ρ be a ranking of item in I such that $\rho(i) > \rho(j)$ if and only if i is ranked higher than j. We can define a value function $V^{\Psi}(\rho)$ that measures how consistent is the ranking ρ with respect to the preference function Ψ as follows:

$$V^{\Psi}(\rho) = \sum_{i,j:\rho(i)>\rho(j)} \Psi(i,j) \tag{7}$$

Therefore, our goal is to produce a ranking ρ^* that maximizes the above value function.

4.3 Greedy Order Algorithm

One possible approach to solve the item ranking problem is to search through the possible rankings in an attempt to find the optimal ranking ρ^* that maximizes the value function defined Equation 7 . However, Cohen et al.[5] showed that finding the optimal ranking ρ^* is a NP-Complete problem based on reduction from the Cyclic-Ordering problem[7] and proposed an efficient greedy order algorithm for finding an approximately optimal ranking shown in Algorithm 1 below:

Algorithm 1 Greedy Order

```
INPUT: an item set $I$; a preference function $\Psi$

OUTPUT: a ranking $\hat{\rho}$

1: for each $i \in I$ do
2:     $\pi(i) = \sum_{j \in I} \Psi(i,j) - \sum_{j \in I} \Psi(j,i)$
3: end for
4: while $I$ is not empty do
5:     $t = \arg\max_{i \in I} \pi(i)$
6:     $\hat{\rho}(t) = |I|$
7:     $I = I - \{t\}$
8:     for each $i \in I$ do
9:         $\pi(i) = \pi(i) + \Psi(t,i) - \Psi(i,t)$
10:    end for
11: end while
```

The algorithm maintains for each item $i \in I$ a *potential* value $\pi(i)$, which is equal to $\sum_{j \in I} \Psi(i,j) - \sum_{j \in I} \Psi(j,i)$. So the more items that are less preferred than i (i.e. $\Psi(i,j) > 0$) the higher the potential of i. The greedy algorithm produces the ranking from the highest position to the lowest position by always picking the item t that currently has the maximum potential and assign it a rank equal to the number of remaining items in I so that it will be ranked above all the other remaining items. It then deletes t from I and updates the potential values of the remaining items by removing the effects of t. The algorithm has a time complexity $O(n^2)$, where n denotes the number of items and it was shown in [5] that the ranking $\hat{\rho}$ produced by the greedy order algorithm has a value $V^{\Psi}(\rho)$ that is within a factor of 2 of the optimal, i.e., $V^{\Psi}(\hat{\rho}) \geq \frac{1}{2}V^{\Psi}(\rho^*)$.

4.4 Random Walk Model for Item Ranking

In this section, we describe a random walk model for ranking the set of items. Instead of directly searching for a ranking as the greedy order algorithm, we attempt to define a Markov chain model in which states correspond to the items and the transitional probabilities depend on a user's preference function Ψ. The stationary distribution of this Markov chain can then be used to produce a ranking.

4.4.1 Random Walk based on User Preferences

The motivation for using the Markov chain model is that it is an effective model for aggregating partial and incomplete preference information from many users. Suppose that among the set of users, some rated i higher than j and others rated j higher than k but very few have rated all three items i,j and k so the preferences regarding i and k is implicit and need to be inferred through transitivity. Such implicit relationships between pairs of items not both explicitly rated by the user could be effectively inferred using multi-step random walks that can exploit the connectivity of the directed graph model underlying the Markov chain.

Our model is closely related to the "PageRank" scheme [3], which defines a random walk on the Web pages based on the Web's hyperlink structure. In particular, the "PageRank" model assumes that a random surfer always randomly pick a hyperlink on the current page to follow at each step. It interprets a direct link from page p to q as an endorsement of q by p so that the stationary probability of being at some page in the long run could reflect its authority. Similarly, our model could be viewed as deriving implicit links between items based on the observed preference information so that a less preferred item j would link to a more preferred item i and the transition probability $p(i|j)$ would depend on the strength of the preference which can be told from the value $\Psi(i,j)$.

Our goal is to obtain a probability distribution $\pi(i)$ over items in I, which could be interpreted as the probability that a user would be interested in each item i. Imagine a user trying to find his favorite item and suppose that he has preferences over the items in his mind. So he would randomly select the items in the following manner. He first chooses an item j randomly. Then based on his preferences, he would switch to another item i based on the conditional probability $p(i|j)$ which would be higher for those items that are more preferred than j and lower for the items that are less preferred than j by the user. Such a process continues and in the long run, the user should select his favorite items most often, which explains intuitively how the stationary distribution could be used to rank the items based on preferences. More precisely, the transition probability $p(i|j)$ of switching to another item j given the current item i is defined as:

$$p(j|i) = \frac{e^{\Psi(j,i)}}{\sum_{j \in \mathcal{I}} e^{\Psi(j,i)}} \tag{8}$$

where the transitional probability $p(j|i)$'s is proportional to $e^{\Psi(j,i)}$.

4.4.2 Compute the Item Rankings

The algorithm for computing the item rankings using the random walk model described in the last section can be described clearly using matrix notations. Let $\mathbf{P}$ be the transition matrix for the random walk model where each entry $p_{i,j}$ is equal to the transition probability $p(j|i)$. Let $\boldsymbol{\pi}_t = [p_t(1), p_t(2), \ldots, p_t(n)]^T$ where $p_t(i)$ is the probability of being at item i after walking t steps. Given an initial probability distribution over the items $\boldsymbol{\pi}_0$, $\boldsymbol{\pi}_t$'s can be computed iteratively using the formula:

$$\boldsymbol{\pi}_t^T = \boldsymbol{\pi}_{t-1}^T \mathbf{P} \tag{9}$$

which is the power or vector iteration methods for solving principal eigenvector problems[9]. The vector of stationary probabilities is defined to be $\boldsymbol{\pi}^* = \lim_{t\to\infty} \boldsymbol{\pi}_t$. In general, using the iterative power method, $\boldsymbol{\pi}_t$ would converge to the dominant or principal eigenvector of the transition matrix $\mathbf{P}$[3]. The existence and uniqueness of the stationary distribution $\boldsymbol{\pi}^*$ is guaranteed if and only if the matrix $\mathbf{P}$ is irreducible and different modifications to $\mathbf{P}$ have been proposed in order to improve numerical stability of the PageRank algorithm[3]. In our model, since the transition probabilities $p(j|i)$'s are calculated using Equation 8, the entries of $\mathbf{P}$ are all non-negative, which could guarantee the existence and uniqueness of the stationary distribution $\boldsymbol{\pi}^*$[1].

4.4.3 Personalization Vector

To avoid the reducibility of the stochastic matrix, Brin and Page[3] proposed a trick which forms a revised transition matrix $\overline{\mathbf{P}}$ by interpolating $\mathbf{P}$ with a perturbation matrix $\mathbf{E} = \mathbf{e}\mathbf{v}^{\mathbf{T}}/n$ where $\mathbf{e}$ is the vector with all components equal to 1 and $\mathbf{v}$ is a "personalization" vector:

$$\overline{\mathbf{P}} = \epsilon \cdot \mathbf{E} + (1-\epsilon) \cdot \mathbf{P} \tag{10}$$

where ϵ is a scalar between 0 and 1. The reasoning is that a web surfer can sometimes "teleport" to other pages according to the probability distribution defined by $\mathbf{v}$ independent of the current page and the parameter ϵ controls how often the surfer may teleport to another page rather than following the hyperlinks. A few follow up works to PageRank proposed different methods to bias the personalization vector $\mathbf{v}$ to take into account different types of information besides the link structure such as contents[10, 23] and user preferences[14].

In our random walk model for item ranking, we follow the similar idea to define a personalization vector $\mathbf{v}^u = [p^u(1), \ldots, p^u(n)]^T$ for each target user u based on his known ratings on the items:

$$p^u(i) = \begin{cases} \frac{e^{r_{u,i} - \overline{r}_u}}{1 + \sum_{i \in I_u} e^{r_{u,i} - \overline{r}_u}} & \text{if } i \in I_u \\ \frac{1}{n - |I_u|} \cdot \frac{1}{1 + \sum_{i \in I_u} e^{r_{u,i} - \overline{r}_u}} & \text{otherwise} \end{cases} \tag{11}$$

So the user would teleport to those items with high ratings more often than to those items that have been rated low by him and all the unrated items have equal probabilities of being visited via teleportation. The use of the personalization vector provides an effective way of incorporating the known preference information of the target user into the random walk model so that the user's given preferences are not only used in selecting the neighbors but also in producing the item rankings.

5. EXPERIMENTS

We evaluated the proposed approach using two sets of real world movie ratings data and conducted different experiments to address the following issues: (1) Does the proposed ranking-oriented CF approach improve item ranking effectiveness compared with traditional rating-oriented approaches such as user-based and item-based algorithms? (2) Which of the two ranking-oriented CF algorithms, namely greedy order and random walk model, is more effective? (3) Is the ranking-oriented similarity measure Kendall Rank Correlation Coefficient more effective in finding users with similar preferences?

5.1 Data Sets

We evaluated the algorithm using two movie ratings data sets: EachMovie[1] and Netflix[2]. The EachMovie data set consists of about 2.8 million ratings made by more than 72 thousand users on 1628 movies. The Netflix data set contains over 100 million ratings from over 480 thousand users on around 18000 titles. The ratings for EachMovie are on a scale from 1 to 6 while the Netflix ratings are on a scale from 1 to 5. Due to the huge size of the Netflix data, we extract a subset comprised of the ratings on the 2000 movies with the most ratings. Then from both data sets, we randomly picked a set of 10600 users that have rated more than 40 different movies, where 10000 are used as the training data and the other 100 and 500 are used for parameter tuning and testing purposes respectively. The density of user-item rating matrix (i.e. proportion of non-zero entries) of the EachMovie and Netflix data sets are 6.1% and 6.6% respectively.

5.2 Evaluation Metric

The major criterion for evaluating traditional rating-oriented collaborative filtering algorithms is the rating prediction accuracy. Commonly used measures for accuracy include the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE), both of which depend on difference between true rating and predicted rating. However, since the emphasis of our work is on improving item rankings instead of rating prediction, we employ the Normalized Discounted Cumulative Gain(NDCG)[13] metric, which is an increasingly popular metric for evaluating ranked results in information retrieval where the documents are assigned graded rather than binary relevance judgements. For collaborative filtering applications, the ratings on items assigned by the users can naturally serve as the graded relevance judgements.

The NDCG metric is evaluated over some number k of the top items on the ranked item list. Let Q be the set of users used for testing and $R(u, p)$ be the rating assigned by u to the item at the p-th position on the ranked list produced for user u. The NDCG at the k-th position with respect to the

[1]http://nyc.lti.cs.cmu.edu/IRLab/11-743s03/lebanon/IR-lab.htm

[2]http://www.netflixprize.com

set of users Q is:

$$\text{NDCG}(Q,k) = \frac{1}{|Q|}\sum_{u\in Q} Z_u \sum_{p=1}^{k} \frac{2^{R(u,p)}-1}{\log(1+p)}$$

where Z_u is a normalization factor calculated so that the NDCG of the optimal ranking has a value of 1. The value of NDCG ranges from 0 to 1 with a higher value indicates better ranking effectiveness. The NDCG is very sensitive to the ratings of the highest ranked items. This is modeled by the discounting factor $\log(1+p)$, which increases with the position in the ranking. This characteristic makes it highly desirable for measuring ranking quality in recommender systems as most users rarely look past the first few items on a recommendation list so the relevance of items at the top positions are far more important than those at low positions.

Note that the Kendall Rank Correlation Coefficient can also be used to evaluate the quality of a ranking by calculating its correlation with the optimal ranking, however such an approach is not able to take into consideration the different importance of high and low positions in a ranking as NDCG does. Therefore, we did not use it as the evaluation metric in our experiments.

5.3 Evaluation Protocol

For each user in the test set, we use 50% of his known ratings as input for model construction and use the remaining 50% as hold-out data for evaluation purposes. Different algorithms are evaluated based on the quality of the rankings they produced for the items in the hold-out data of each user, which can be measured using NDCG based on the true ratings on the items.

5.4 Impact of Parameters

5.4.1 Impact of Neighborhood Size

Similar to user-based and item-based collaborative filtering algorithms, the size of the neighborhood will affect the performance of our algorithms which need to estimate a user preference based a set of neighbors. For this set of experiments, we fix the number of training users at 5000 and use NDCG at the 1st position as the performance measure. We run both the greedy order algorithm and the random walk algorithm with varying neighborhood size. Figure 1 shows the change in performances when the size of the neighborhood increase from 20 to 200. We can see that the NDCG gradually increases as the neighborhood size increases from 20 to 100 since the preference $\Psi(i,j)$ can be estimated more accurately given more neighbors. However, we also observe that the performance start to decrease as the neighborhood size exceeds 100, which is due to that many non-similar users began to enter the neighborhood and introduce incorrect preference information into $\Psi(i,j)$ as the neighborhood size gets too large. The optimal value for the neighborhood size is around 100.

5.4.2 Impact of ϵ

The parameter ϵ in Equation 10 controls how often the "teleport" operation is performed in the random walk model. By setting ϵ to 0, the transition probabilities will be defined by Equation 8, which is only determined by the preference function $\Psi(i,j)$. On the other hand, by setting ϵ to 1, the random walk model will perform the "teleport" operation

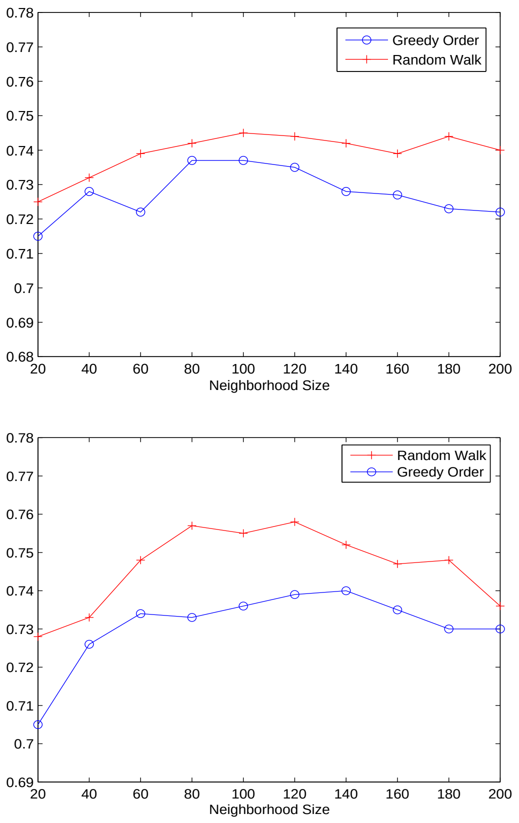
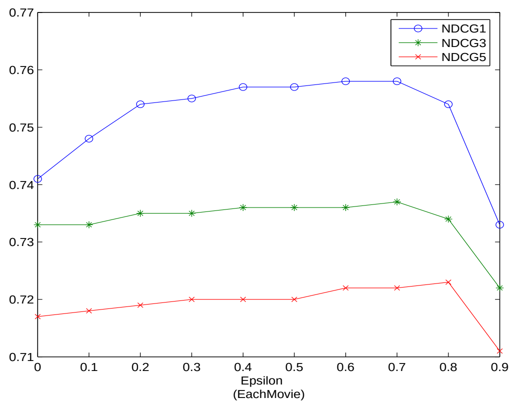


Figure 1: NDCG vs. Neighborhood Size Results on EachMovie (Above) and Netflix (Below)

all the time and the preference information $\Psi(i,j)$ is totally ignored. In our experiments, we vary ϵ from 0 to 0.9 and measure the NDCG at the 1st, 3rd and 5th positions denoted by NDCG1, NDCG3 and NDCG5 respectively. The results are shown in Figure 2. We can see that increasing the ϵ value from 0 can generally lead to improvements on all three NDCG measures and the improvements on NDCG1 are especially notable. We also note that as ϵ gets very large the performances would start to drop significantly. This is because with high ϵ value, the random walk model would perform the "teleport" operation most of the time and the stationary distribution would approach the personalization vector defined in Equation 11, which assigns equal probabilities to all the unrated items and makes it impossible to rank the unrated items effectively. The optimal value for ϵ is around 0.6.

5.5 Comparisons with other Algorithms

We choose 4 rating-oriented collaborative filtering algorithms as the baselines including item based model using Pearson Correlation Coefficient(IPCC) and Vector Similarity(IVS), user based model using Pearson Correlation Coefficient (UPCC) and Vector Similarity(UVS)and compared them with our algorithms: random walk model using Pearson Correlation Coefficient (RWPCC), Vector Similarity (RW VS), and Kendall Rank Correlation Coefficient (RWKRCC), greedy order algorithm using Pearson Correlation Coefficient(GOPCC), Vector Similarity (GOVS) and Kendall Rank



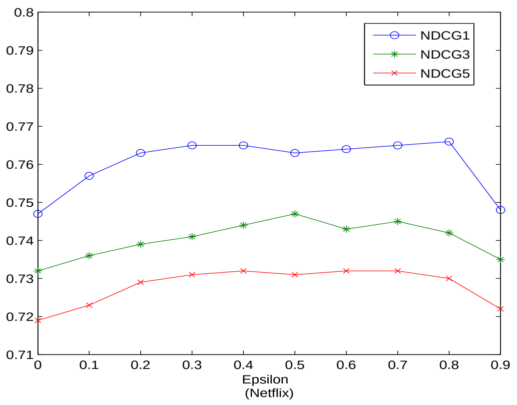


Figure 2: NDCG vs. Epsilon Results on EachMovie (Above) and Netflix (Below)

Correlation Coefficient (GOKRCC). For IPCC and IVS, we set the number of neighboring items to be 50 as suggested by [24]. For all the other algorithm, we set the size of the neighborhood N_u to be 100. The parameter ϵ for the random walk based models are set to 0.6.

The evaluation results on both EachMovie and Netflix data sets are shown in Table 1(a) and 1(b) respectively. For both data sets, we vary the number of training users from 1,000 to 10,000. For each algorithm, we report the NDCG values at the 1st, 3rd and 5th positions, denoted by NDCG1, NDCG3 and NDCG5 respectively. For each column in Table 1(a) and 1(b), we have highlighted the top 3 performers and the values shown in the bottom row are the performance improvements achieved by the best ranking-oriented methods over the best rating-oriented methods.

As can be seen from the results, in all the conducted experiments, the top 3 performers are consistently ranking-oriented methods. The improvements on NDCG1, NDCG3 and NDCG5 of the best ranking oriented methods over the best rating oriented methods are more than 8.8%, 4.7% and 3.4% on average. We can see that random walk model had consistently outperformed all the rating oriented methods and the greedy order algorithm in almost all the experiments.

For the greedy order algorithm, we can see that using Kendall Rank Correlation Coefficient as the similarity measure has led to higher performances than the Pearson Correlation Coefficient and the Vector Similarity on both data sets. For the random walk model, the performances of RW-PCC and RWKRCC are very close on the EachMovie data set while RWKRCC outperformed RWKRCC on the Netflix data. So from the results we can see that the Kendall Rank Correlation Coefficient is a more effective similarity measure for finding users with similar preferences. It can also be noted that the performances of different algorithms would improve as the number of training users increases. This is because with a larger user database, it will be easier to find sufficient number of neighbors with high similarities to the target user so that his preferences can be estimated more accurately.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a ranking-oriented framework for collaborative filtering that addresses the item ranking problem directly without trying to predict a user's ratings on the unrated items as an intermediate step. Our approach extends the neighborhood-based collaborative filtering framework by identifying and aggregating the preferences rather than ratings of similar users in order to produce a ranking of items. We described two methods for computing the item rankings based on preferences including a greedy order algorithm and a random walk model. Experimental results show that our approach outperforms existing CF algorithms in terms of ranking effectiveness.

For future work, we would like to investigate different techniques proposed for improving traditional rating oriented CF including data smoothing and utilizing content information, and study their usefulness to our ranking-oriented CF approach.

7. ACKNOWLEDGEMENTS

The authors are supported by a grant from NEC Labs China and Hong Kong RGC project number 621307.

8. REFERENCES

[1] M. Bianchini, M. Gori, and F. Scarselli. Inside pagerank. *ACM Trans. Internet Technology*, 5(1):92–128, 2005.

[2] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of UAI 1998*, pages 43–52, 1998.

[3] S. Brin and L. Page. Anatomy of a large-scale hypertextual web search engine. In *Proceedings of 7th International World Wide Web Conference*, 1998.

[4] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of ICML 2005*, pages 89–96, 2005.

[5] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 5:243–270, 1999.

[6] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *Proceedings of ICML 1998*, pages 170–178, 1998.

[7] M. R. Gary and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

Table 1: Comparison between Ranking and Rating-Oriented CF algorithms

	1000 Training Users			5000 Training Users			10000 Training Users		
Methods	NDCG1	NDCG3	NDCG5	NDCG1	NDCG3	NDCG5	NDCG1	NDCG3	NDCG5
UPCC	0.690	0.680	0.691	0.706	0.693	0.702	0.700	0.693	0.698
UVS	0.679	0.675	0.680	0.669	0.676	0.685	0.677	0.687	0.689
IPCC	0.652	0.628	0.640	0.670	0.651	0.662	0.676	0.663	0.671
IVS	0.611	0.607	0.610	0.607	0.621	0.635	0.652	0.628	0.640
GOPCC	**0.736**	0.693	0.687	**0.741**	0.708	0.706	0.733	0.708	0.711
GOVS	0.698	0.680	0.683	0.711	0.697	0.690	0.690	0.696	0.686
GOKRCC	0.728	**0.704**	**0.706**	0.740	0.713	**0.716**	**0.745**	**0.720**	**0.720**
RWPCC	**0.736**	**0.705**	**0.706**	**0.768**	**0.727**	**0.724**	**0.780**	**0.734**	**0.729**
RWVS	0.701	0.684	0.689	0.740	**0.715**	0.710	0.719	0.711	0.705
RWKRCC	**0.748**	**0.709**	**0.706**	**0.768**	**0.728**	**0.728**	**0.771**	**0.730**	**0.731**
	8.4%	4.2%	2.1%	8.7%	5.1%	3.7%	11.4%	5.9%	4.7%

(a) Results on EachMovie Data Set

	1000 Training Users			5000 Training Users			10000 Training Users		
Methods	NDCG1	NDCG3	NDCG5	NDCG1	NDCG3	NDCG5	NDCG1	NDCG3	NDCG5
UPCC	0.669	0.676	0.679	0.657	0.683	0.689	0.679	0.693	0.699
UVS	0.671	0.679	0.685	0.706	0.708	0.711	0.713	0.714	0.718
IPCC	0.652	0.647	0.649	0.699	0.682	0.687	0.716	0.701	0.700
IVS	0.625	0.639	0.647	0.671	0.670	0.674	0.678	0.677	0.683
GOPCC	0.715	0.699	0.697	0.743	0.718	0.716	0.732	0.718	0.717
GOVS	0.700	0.700	**0.704**	0.732	0.725	0.724	0.739	0.726	0.726
GOKRCC	0.718	**0.704**	**0.704**	**0.750**	**0.728**	**0.726**	**0.748**	**0.732**	**0.729**
RWPCC	**0.727**	0.698	0.698	**0.762**	**0.728**	0.721	**0.752**	0.730	0.726
RWVS	**0.727**	**0.713**	**0.711**	**0.750**	0.727	**0.723**	0.743	**0.733**	**0.727**
RWKRCC	**0.732**	**0.715**	**0.712**	**0.765**	**0.738**	**0.732**	**0.765**	**0.742**	**0.738**
	9.1%	5.3%	3.9%	8.3%	4.2%	2.9%	6.8%	3.9%	2.7%

(b) Results on Netflix Data Set

[8] K. Y. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[9] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins University Press, London, 1996.

[10] T. H. Haveliwala. Topic-sensitive pagerank. In *Proceedings of WWW 2002*, 2002.

[11] J. Herlocker, J. A. Konstan, and J. Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information retrieval*, 5(4):287–310, 2002.

[12] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.

[13] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[14] G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of WWW 2003*, pages 271–279, 2003.

[15] R. Jin, L. Si, C. Zhai, and J. Callan. Collaborative filtering with decoupled models for preferences and ratings. In *Proceedings of CIKM 2003*, pages 309–106, 2003.

[16] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of SIGKDD 2002*, pages 133–142, 2002.

[17] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: Applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, 1997.

[18] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[19] H. Ma, I. King, and M. R. Lyu. Effective missing data prediction for collaborative filtering. In *Proc. of SIGIR 2007*, pages 39–46, 2007.

[20] J. I. Marden. *Analyzing and Modeling Rank Data*. Chapman & Hall, New York, 1995.

[21] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach. In *Proc. of UAI*, pages 473–480, 2000.

[22] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proc. of ACM Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.

[23] M. Richardson and P. Domingos. The intelligent surfer: Probabilistic combination of link and content information in pagerank. In *Proceedings of NIPS 2001*, pages 1441–1448, 2001.

[24] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.

[25] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proc. of SIGIR 2005*, pages 114–121, 2005.