CMAP: Effective Fusion of Quality and Relevance for Multi-criteria Recommendation

Xin Xin, Michael R. Lyu
Dept. of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
{xxin,lyu}@cse.cuhk.edu.hk

Irwin King*
AT&T Labs – Research
180 Park Ave
Forlham Park, NJ 07932
irwin@research.att.com

ABSTRACT

The research issue of recommender systems has been treated as a classical regression problem over the decades and has obtained a great success. In the next generation of recommender systems, multi-criteria recommendation has been predicted as an important direction. Different from traditional recommender systems that aim particularly at recommending high-quality items evaluated by users' ratings, in multi-criteria recommendation, quality only serves as one criterion, and many other criteria such as relevance, coverage, and diversity should be simultaneously optimized. Although recently there is work investigating each single criterion, there is rarely any literature that reports how each single criterion impacts each other and how to combine them in real applications. Thus in this paper, we study the relationship of two criteria, quality and relevance, as a preliminary work in multi-criteria recommendation. We first give qualitative and quantitative analysis of competitive quality-based and relevance-based algorithms in these two criteria to show that both algorithms cannot work well in the opposite criteria. Then we propose an integrated metric and finally investigate how to combine previous work together into an unified model. In the combination, we introduce a Continuous-time MArkov Process (CMAP) algorithm for ranking, which enables principled and natural integration with features derived from both quality-based and relevance-based algorithms. Through experimental verification, the combined methods can significantly outperform either single quality-based or relevance-based algorithms in the integrated metric and the CMAP model outperforms traditional combination methods by around 3%. Its linear complexity with respect to the number of users and items leads to satisfactory performance, as demonstrated by the around 7-hour computational time for over 480k users and almost 20k items.

Categories and Subject Descriptors

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

*Irwin King is currently on leave from the Chinese University of Hong Kong, where the major part of this research was performed.

WSDM'11, February 9–12, 2011, Hong Kong, China.

General Terms

Algorithms, Experimentation

Keywords

Collaborative Filtering, Continuous-time Markov Process, Recommender System

1. INTRODUCTION

With the explosive growth of resources on the Web, recommender system is to suggest items (products, movies, etc.) according to users' past behaviors, in order to filter information. Usually, the quality of items is rated by users. For example, the ratings can be a score from 1 to 5 with higher value indicating the better quality. During the recent decades, the research issue of recommender systems has been formulated as a classical regression problem of predicting the quality ratings and has been explored deeply by both research and industry community [1, 5]. Typical approaches include factorization models and neighborhood models. The former focus on a global view of all users and items for predicting user's ratings on different items; and on the other hand, the latter concern a local view of the nearest similar users and items. Among the hybrid methods, the work of [17] has been demonstrated the most effective in a well refined Netflix competition. Other competitive work include [20, 28, 36, 39].

In the next generation of recommender systems, multi-criteria recommendation has been predicted as an important direction [1]. The concept "multi-criteria" here refers to some other recommendation criteria besides quality evaluated from ratings. Typical criteria include relevance, coverage, diversity, etc. For example, the work of [7] and the tasks in the KDD-cup 2007 [18, 27, 33] emphasize on optimizing for recommending items with more relevance or user interests, which are relevance-based algorithms. Relevance is a different criterion from the criterion of quality. For example, in movies recommendation, quality refers to a user's evaluation on a movie's plot, acting, special effects; while relevance refers to a user's interests to see a movie. In books recommendation, quality refers to a book's content worthiness; while relevance refers to a book's attractiveness to a user. A user may give a high rating to a classical movie for its good quality, but he/she might be more likely to watch a recent one that is more relevant and interesting to their lives, though the latter might be worse in quality. Different from quality-based recommendation that focuses on recommending items that will likely to obtain high ratings from users, relevance is reflected by whether a user will hit (or visited/rated) an item and therefore, relevance-based recommendation focuses on recommending items that will be likely to be hit by a user in the future. Coverage and diversity are other criterion examples. Cov-

erage means to what extend the recommendation can cover all the items [34]; and diversity means how different the recommended items are from each other [11]. Although one may argue that the evaluation of relevance, coverage, and diversity might also be contained in ratings, from previous work [7, 15], ratings reflect mainly for quality rather than other criteria. A recommender system's success in multi-criteria recommendation should consider all the criteria besides quality.

The concept of multi-criteria recommendation is also different from most ensemble ideas in recent work. The ensemble in recent literatures is mainly refer to utilize more features for optimizing a single criterion. For example, in the work of [17], both ratings (explicit feature) and hits history (implicit feature) are utilized to optimize the criterion of quality; and in the work of [7], both user behavior features and content features are utilized to optimize the criterion of relevance. But in multi-criteria recommendation, the optimization goal should combine multiple criteria. It should simultaneously optimize all the factors of quality, relevance, coverage, and diversity.

Intuitively some conflicts might happen from different criteria, thus in multi-criteria recommendation it is important to investigate how each criterion impacts each other, how to combine multiple criteria and how to combine previous algorithms together. For example, if a user's neighbor has given a very low rating to a movie A, in quality-based algorithms, A might not be recommended to this user because it is likely to obtain low rating indicated from his/her neighbor's rating records; in relevance-based algorithms, however, A might be recommended because the neighbor at least has shown interest to see it. Although multi-criteria recommendation is obviously practical in real applications and might even have already been integrated in many online systems, to the best of our knowledge, there is rarely any reports in literature on these research issues, which motivates our work in this paper.

In this paper, we study the interplay relationship of different criteria and investigate how to combine previous single criterion approaches in multi-criteria recommendation. Specifically, we investigate the criteria of quality and relevance as a preliminary work because they are more practical and both have enough previous work. We give both qualitative and quantitative analysis of competitive quality-based and relevance-based algorithms in these two criteria. As the first ever solution for considering both quality and relevance in multi-criteria recommendation, we propose a combined metric and investigate methods to fuse competitive quality-based and relevance-based algorithms together. In the fusing, fundamental combination methods suffer from the integration-unnatural and quantity-missing problems. To address this limitation, we propose to extend the random walk theory from previous work of Discrete-time Markov Process (DMP) to Continuous-time MArkov Process (CMAP) for combining additional features. Consequently, the combination is an unified models having an intuitive interpretation with quantity information retained; and the accuracy is better than baseline methods through experiment justification. The CMAP algorithm scales linearly with the number of users and items. Thus it can be employed in applications with large-scale data.

The main contributions in this paper lie in that we justify that both quality-based algorithms and relevance-based algorithms cannot work well in the opposite criteria through both qualitative and quantitative analysis. We propose an integrated metric and introduce a large scalable framework CMAP to fuse the criteria of quality and relevance in multi-criteria recommendation. It can significantly outperform single quality-based and relevance-based algorithms in the integrated metric and can outperform traditional combination methods by around 3% by empirical study on two real world datasets. We hope that such work will be helpful from a technical standpoint in improving real recommendation applications.

The remainder of this paper is organized as follows. In Section 2, we introduce our analysis on the relationship of quality criterion and relevance criterion in recommendation. In Section 3, we describe an integrated metric proposed for considering multi-criteria. In Section 4, we investigate how to combine previous work into an unified model in order to optimize integrated metric. Experimental results and analysis are shown in Section 5. Related work is presented in Section 6, and we conclude the paper in Section 7.

2. QUALITY VS. RELEVANCE

In this section, we study the relationship of quality criterion and relevance criterion in current recommender systems. The goal is to investigate whether quality-based algorithms can achieve good relevance performance and whether relevance-based algorithms can achieve good quality performance. To verify this, we conduct both quantitative and qualitative analysis. We mainly show the qualitative analysis for an intuitive impression first and will detail the quantitative analysis in experimental section.

The qualitative analysis is designed as follows. We study the data of Netflix[1], a famous large-scale dataset for movie recommendation. Since qualitative analysis should focus on the whole view, we utilize an item's average rating score to describe its quality and an item's hitting count to describe its relevance. The assumption is that if an item is well evaluated by many users, it is a high-quality item to most users; and if an item is visited by many users, it is a high-relevance item to most users. Fig. 1 shows the distribution of all 17,770 items on the measure of quality and relevance. From this figure, we can observe that there are four types of items: A) normal-quality and high-relevance; B) high-quality and high relevance; C) normal-quality and normal-relevance; and D) high-quality and normal-relevance. Typical example of each type is shown in the right part of the figure. A success recommendation should contain type A B and D. We choose user/item-based Pearson Correlation Coefficient (PCC) [4, 29], Aspect Model (AM) [16], PMF [28] and EigenRank [20] as quality-based algorithms; and within relevance-based algorithms, an association-based method [7] and a hitting-frequency-based method [33] are chosen. We randomly choose 40,000 users for training and 10,000 users (given their first 10 ratings) for testing. We make statistics of the top five recommended items by both kinds of algorithms. Fig. 2 shows the distributions of these items on the measure of quality and relevance. Due to space limitation, only results of EigenRank (quality-based) and the association-based method (relevance-based) are shown as representatives; and similar results can be obtained for other methods within the same type. In this figure, the color of a small area denotes the log value of the total recommended items' occurrence count within the quality and relevance metrics.

It can be obtained that for quality-based algorithms, most recommended items belong to Type B and D; items of Type A are almost missing. For relevance-based algorithms, most recommended items belong to Type A and B; and items of Type D are almost missing. In quantitative analysis, the same problem can be found, which will be detailed in experimental section. Thus from these analysis, we can conclude that both quality-based methods and relevance-based methods cannot perform well in the opposite criteria. Thus both quality-based and relevance-based methods are recommending incomplete items.

But the missing items are important in recommendations. Items of Type A are especially concerned by commercial companies in

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

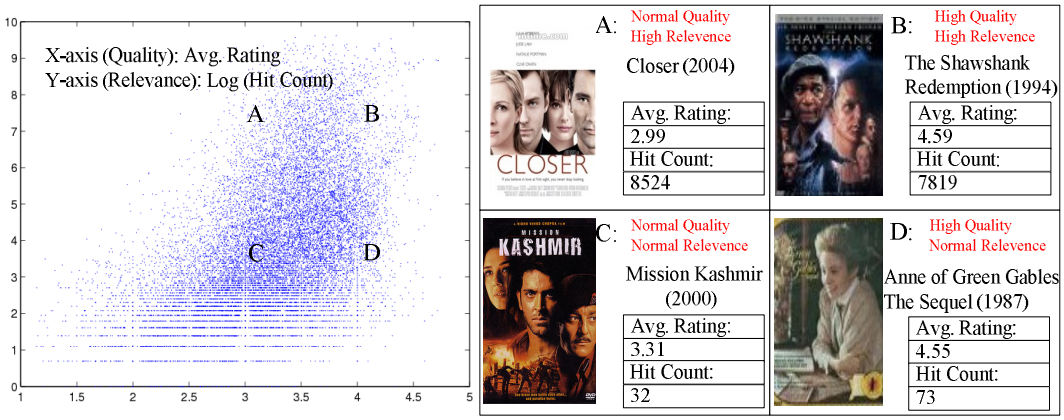


Figure 1: Distribution of items in relevance and quality

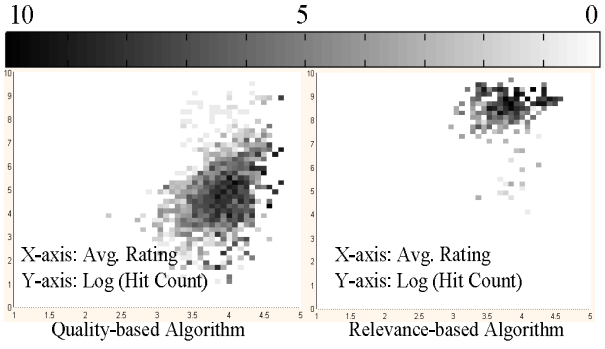


Figure 2: Distribution of recommended results

advertising or selling business [30], because it reflects the predictions of hitting counts or sales, which directly influence their revenue. Items of Type D, on the other hand, have been demonstrated valuable in recent long tail research [9]. These high-quality items are only attractive to a limited number of particular users; but if we cumulate the effect of all these items, great potential can be explored [10]. Therefore, items in both Type A and D are important and should not be ignored. In addition, for most users, for cases that two items with the same quality, the more relevant one will be more likely to satisfy most users, and so are the opposite cases. However, in current recommendation methods, quality-based algorithms miss the factor of relevance. Consequently, users may not show interests to visit some of the recommended items. Relevance-based algorithms, on the other hand, miss the factor of quality. Thus users will suffer from normal-quality recommended results. This is also the reason why multi-criteria recommendation is important.

3. INTEGRATED METRIC

For considering both quality and relevance for multi-criteria recommendation, we linearly combine quality-based and relevance-based metrics as an integrated evaluation measure. Quality-based metrics measure the closeness of a recommender system's predicted ratings to the users' real ratings. Classical metric is Mean Absolute Error (MAE), root mean squared error (RMSE), etc. Relevance-based metrics, on the other hand, measure the likelihood that an item will be hit. Classical metrics include precision/recall, Receiver Operating Characteristic (ROC) curve, etc. In application systems, a rank list is the final output for users. Thus normalized discount cumulated gain (NDCG) has been recently employed to evaluate recommendation results including both quality-based NDCG [20] and relevance-based NDCG [13]. In fact, there are many ways for the integrated metric; however, the comparisons of metrics are beyond the scope of this paper. We specifically propose the following NDCG metric as a first ever solution. We are also awared that NDCG is not the most widely used metric in recommender systems, but there are two supportive reasons for this: 1) NDCG is a ranking-oriented metric, which is more practical in application systems. Comparing other ranking-oriented metrics, NDCG is a position dependent metric. It assigns top positions more weight, which is more reasonable. 2) Both quality-based NDCG and relevance-based NDCG are accepted as practical measures in previous work of recommender systems [13, 20]. It is also convenient for the integration because the values have similar meanings in both tasks and are compatible for combination.

Given the rank of recommended results, quality-based NDCG at position P is defined as (referring to [20])

$$NDCG_{P-quality} = \frac{1}{U}\sum_{u}^{U} Z_u \sum_{p=1}^{P} \frac{2^{r_{u,p}}-1}{\log(1+p)}, \quad (1)$$

where U is the number of users, Z_u is a normalization factor of user u, and $r_{u,p}$ is the ground truth rating score by user u on the item at position p. Since only a limited number of items rated by users are selected as ground truth, by following [20], uncertain ones in the rank are removed before calculation. Relevance-based NDCG at position P, on the other hand, is defined as (refering to [13])

$$NDCG_{P-relevance} = \frac{1}{U}\sum_{u}^{U} Z_u \sum_{p=1}^{P} \frac{2^{h_{u,p}}-1}{\log(1+p)}, \quad (2)$$

where $h_{u,p}$ is a binary value function indicating whether user u has hit the item at position p. In this paper, we follow the same idea in [7]: a hit is defined by whether the user has rated the item. In both metrics, NDCG value is scaled from 0 to 1 with higher value indicating better results. We linearly integrate these two metrics as

$$NDCG_{I-linear} = \lambda * NDCG_Q + (1-\lambda) * NDCG_R. \quad (3)$$

λ scales from 0 to 1 and can be set in different applications by particular users. When $\lambda = 0$, it is single relevance-based NDCG; when $\lambda = 1$, it is single quality-based NDCG; in other cases, when λ increases, the evaluation will emphasize more on quality-based performance and when decreases, it will emphasize more on

relevance-based performance. In our experiments, we evaluate performances of recommendation algorithms on all the scale of λ from 0 to 1 with the interval of 0.1. Thus different configurations for the balance from users can be adapted.

4. COMBINED FRAMEWORK

In this section, we introduce our methodology for optimizing the integrated metric. We first present our rationale of basic approaches selection from competitive quality-based and relevance-based methods. Then for the combination, we propose three methods. The first one is LinearComb, a linear combination of different algorithms' values. Nevertheless, as we will show later, features contain incompatible values in different basic methods, thus a linear combination will be unnatural and will make the accuracy decrease [12]. The second method is RankComb, a rank-based integration, which is to combine the rank results of different recommendation algorithms linearly by Borda count [8]. Yet this means the quantity information from recommendation results will be lost. Thus it will also make the accuracy decrease. Therefore, we propose the third method based on CMAP, to solve above problems. By further employing queueing system theory, CMAP has a intuitive interpretation without losing quantity information.

4.1 Rationale of Basic Approaches Selection

Within quality-based algorithms, we choose EigenRank as the fundamental method for its advantage of modeling user preference order directly. Currently, state-of-the-art quality-based methods are divided into rating-value-oriented (e.g., PMF [28]) and ranking-oriented (e.g., EigenRank [20]). Rating-value-oriented methods first predict ratings, and then produce the rank. The disadvantage is that rating values are predicted independently without considering the preference order of two items, thus the accuracy of recommendation will decrease. Ranking-oriented methods, on the other hand, predict the rank of recommended items based on directly modeling the preference order of arbitrary two items "without going through the inter-meditate step of rating prediction" [20]. Thus this kind of method is more effective in practical E-Commerce applications. This model is proven to outperform many classical methods [20]; and we have also demonstrated that it outperform PMF in relevance-based NDCG through experiments. EigenRank model is based on random walk theory, which is a special case of DMP. A stationary distribution of the DMP is employed to decide the preference score of an item. Formally, let $T = \{0, 1, 2, ...\}$ be a discrete time set, and $S = \{1, 2, ..., N\}$ be a state set. The process can be formulated by a stochastic variable sequence $\{X_t, t \in T\}$. For arbitrary $i_0, i_1, ..., i_t, i_{t+1} \in S$, we have

$$P\{X_{t+1} = i_{t+1} | X_0 = i_0, X_1 = i_1, ..., X_t = i_t\} = P\{X_{t+1} = i_{t+1} | X_t = i_t\}. \quad (4)$$

The stationary distribution of this DMP is defined as

$$\pi = \pi * P, \quad (5)$$

where P is probability transition matrix, and π is the stationary distribution vector. P is built as

$$p_{ij} = p(j|i) = \frac{e^{\psi(j,i)}}{\sum_{j \in S} e^{\psi(j,i)}}, \quad (6)$$

where $\psi(i, j)$ is a preference function defined for each user u on two arbitrary items i and j as

$$\psi(i, j) = \frac{\sum_{v \in N_u^{i,j}} s_{u,v} \cdot (r_{v,i} - r_{v,j})}{\sum_{v \in N_u^{i,j}} s_{u,v}}. \quad (7)$$

In this equation, $N_u^{i,j}$ is the set of u's neighbors, and $s_{u,v}$ is the Kendall Rank Correlation Coefficient (KRCC) [24] similarity.

Within relevance-based algorithms, we choose an association-based method [7] and a hitting-frequency-based method [33]. Association and hitting frequency have been demonstrated as two competitive features in relevance-based approaches. Therefore, in our framework, we combine association and hitting frequency into the EigenRank model to form an unified recommendation framework. Association feature describes the number of users who have hit the same two items. The fundamental assumption of this method is that frequent co-occurrence items in the past are also likely to appear together in the future. In other words, if a large number of users hit both Item M and Item N; another user hit only one of them; then he/she is likely to hit the other one. This feature has been demonstrated effective as a state-of-the-art relevance-based algorithm in [7]. Hitting-frequency feature describes an item's recent total hitting count. The fundamental assumption of this method is that popular items are likely to interest users. In other words, for two items, a user is likely to hit the one with more hitting count. This feature has been demonstrated effective in recent relevance-based recommendation competition KDD-cup 2007 [18, 33].

4.2 LinearComb

LinearComb method is to linearly combine the results, which is the most intuitive and direct way for combination. In this method, the final recommendation score ($S_{LinearComb}$) for each item is defined as

$$S_{LinearComb} = w_1 F(EigenRank) + w_2 F(Assoc) + w_3 F(Hit - freq). \quad (8)$$

F is a normalization function to convert different scales of variables into 0 to 1, which is defined as

$$F(x) = 1/(1 + \exp(-x)). \quad (9)$$

w is function weight vector to be tuned for each sub-methods. In the three results to be combined, the score of EigenRank is a stationary probability value ranging from 0 to 1; and the scores of association and hitting-frequency are integer count values from 1 to $maximum$. Thus the values to combine are incompatible, which makes the fusion unnatural. Even we have converted them into the same scale, previous work has also indicated that the accuracy will decrease in such cases [12].

4.3 RankComb

To solve the problem of LinearComb, we propose another fundamental combination method RankComb, which combines the ranks from different methods by employing Borda count [8]. The main advantage of this method is the values in the combination are compatible. Given a recommended rank of an algorithm, RankComb first calculates a Borda count (BC) value for each item defined as

$$BC_{item} = 1/position(item). \quad (10)$$

Then, the item's BC values from different algorithms can be linearly combined as a new recommendation score as

$$BC_{RankComb} = w_1 BC_{EigenRank} + w_2 * BC_{Assoc} + w_3 * BC_{Hit-freq}. \quad (11)$$

The final results will refer to this new BC value. Although this method solve the integration-unnatural problem of LinearRank, during the process of converting a rank to BC values, the quantity information of these results from different methods are missing. This will also make the recommendation accuracy decrease.

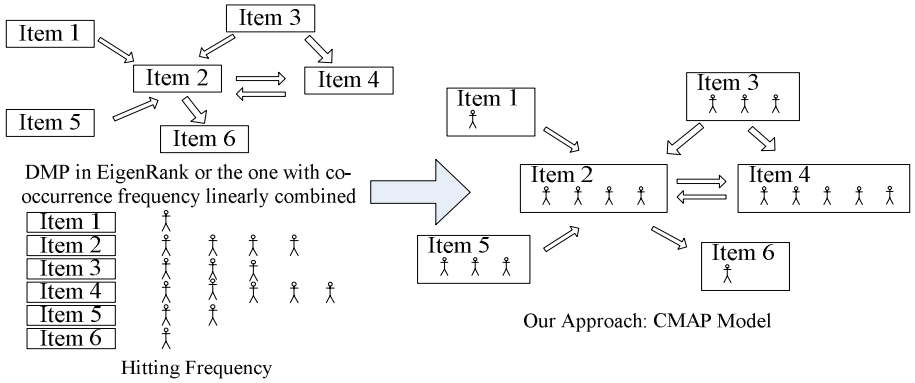


Figure 3: An overview of CMAP framework

4.4 CMAP

To attack the integration-unnatural and quantity-missing problems in fundamental combination methods, we propose the CMAP model, which retains the quantity information and has an intuitive interpretation.

CMAP is a general framework which can integrate both relational and local features. In this work, we show examples of integrating the association feature (a relational feature) and the hitting frequency feature (a local feature). Other features can be integrated into CMAP similarly by linear combination.

4.4.1 Association Feature Combination

Since association is a relational feature of two items, we can employ a similar idea with the random walk process in EigenRank to model and integrate it. The difference is that the transition matrix is determined by association feature instead of rating information. The questions are how to define neighbors and the transition matrix.

In defining neighbors, we utilize cosine-based similarity for simplicity. The similarity of two users u and v is defined as

$$s'_{u,v} = \frac{\overrightarrow{R}_u * \overrightarrow{R}_v}{\|\overrightarrow{R}_u\| * \|\overrightarrow{R}_v\|}, \tag{12}$$

where $\overrightarrow{R}_u$ is the hitting history vector of user u. We set a threshold to select similar users as the current user's neighbors. In building the transition matrix, following the idea in [7], we define a co-occurrence function of two items i and j as

$$\xi(i,j) = \frac{Freq(ij)}{Freq(i) * Freq(j)^{\beta}}, \tag{13}$$

where $Freq(*)$ is the occurrence times and β is a control parameter ranging from 0 to 1. When β equals 0, the formula is the probability of co-occurrence of i and j on the condition of occurrence of i. In this case, these is a limitation that frequent items will obtain excessive bias [4, 7]. Thus the term $Freq(j)^{\beta}$ is added following the idea of inverse document frequency [25] to adjust the weight. Therefore, the transition matrix P' can be defined as

$$p'_{ij} = p'(j|i) = \frac{\xi(i,j)}{\sum_{j \in S} \xi(i,j)}. \tag{14}$$

Since the transition matrix P' includes compatible values to P in EigenRank, we can combine association feature to this model by linear combination as

$$P_{new} = P * \alpha + P' * (1 - \alpha). \tag{15}$$

4.4.2 Hitting Frequency Combination

Different from relational features like previous association feature, hitting-frequency is a local feature and it is difficult to be linearly combined into probability transition matrix. The main problem is that DMP in EigenRank cannot model local features. Thus we propose to extend it to CMAP, by which a new variable will be added to model local features, making the combination have an intuitive interpretation. Different from DMP where the random walk is in discrete steps, in CMAP it is a continuous process. This means the staying time at each state is considered as shown in Fig. 3. Frequent items should have longer staying time. Formally, let $S = \{1, 2, ..., N\}$ denote the state set. The stochastic variable sequence in CMAP is denoted as $\{X_t, t \geq 0\}$. For arbitrary $0 \leq t_0 < t_1 < ... < t_n < t_{n+1}, i_k \in S, 0 \leq k \leq n+1$,

$$P\{X_{t_{n+1}} = i_{n+1} | X_{t_0} = i_0, X_{t_1} = i_1, ..., X_{t_n} = i_n\} = P\{X_{t_{n+1}} = i_{n+1} | X_{t_n} = i_n\}. \tag{16}$$

For simplicity, we assume it is a time-homogenous Markov Process, thus it has the following property

$$P\{X_{s+t} = j | X_s = i\} = P\{X_t = j | X_0 = i\} = p_{ij}(t). \tag{17}$$

CMAP is described by Q-matrix instead of the transition matrix in DMP. Q-matrix is defined as

$$\begin{cases} q_{ij} = p'_{ij}(0) = \lim_{t \to 0} \frac{p_{ij}(t)}{t}, i \neq j; \\ q_{ii} = -\lim_{t \to 0} \frac{1 - p_{ii}(t)}{t}. \end{cases} \tag{18}$$

Under such description, it can be proven [32] that the staying time at each state follows an exponential distribution

$$P(\tau > t | X_{pre} = i) = exp(q_{ii}t), \tag{19}$$

where X_{pre} denotes the previous state and τ is the staying time defined as:

$$\tau = inf\{t : t > 0, X_t \neq X_{pre}\}. \tag{20}$$

It can also be proven [32] that

$$P[X_\tau = j | X_0 = i] = \frac{q_{ij}}{-q_{ii}}. \tag{21}$$

From Eq. (19) and Eq. (21), we obtain formulations of the two determining factors of a CMAP, staying time distribution and transition probabilities. As we have utilized the transition matrix to model relational features before, the staying time distribution, an exponential distribution, is just the one to model local feature of hitting frequency.

Algorithm 1 CMAP Algorithm

Inputs: Rating information of current user and users in training set
Outputs: The ranked list of unrated items for each user as the recommendation results

1: Estimate q_{ii} for each item i based on recent hitting count of each item according to Eq. (24)
2: **for** each user **do**
3: Calculate KRCC and cosine similarities between current user and users in training set
4: Select its neighbors based on the similarities
5: Build probability transition matrix from Eq. (15)
6: Calculate stationary distribution of CMAP according to Eq. (25)
7: Rank the items based on the probabilistic score of a stationary distribution of CMAP
8: Remove the rated items
9: **end for**

We propose to utilize the following formulation to model local feature by employing queueing theory. Because it makes the staying time have a practical meaning of waiting time in a queueing system in addition to its effectiveness in accuracy. As shown in the bottom-left part in Fig. 3, we suppose that there is a ticket selling window for each item, and the users who recently hit the item are costumers buying tickets. We assume that the temporal sequence of the costumers' arrival follows the time-homogenous Poisson Process, with v as the costumer-arriving rate. The services at the ticket selling windows have the same speed to process a deal. The time for each deal follows an exponential distribution with the same service rate u. To make the system stable, we set $u > v$. In such a queueing system, for each item, let T_g denote the waiting time of a customer buying its ticket, then it can be concluded [2] that

$$P(T_g \leq x) = 1 - \frac{v}{u} exp^{-(u-v)x}. \tag{22}$$

Specifically, on the condition that there is a queue, we fortunately obtain an exponential distribution that fits the requirement.

$$P(T_g \leq x | the\ queue\ exists) = 1 - exp^{-(u-v)x}. \tag{23}$$

Thus we propose to model hitting frequency using

$$q_{ii} = -(u - v_i). \tag{24}$$

If u is larger, the variance of waiting time conditions becomes smaller, which means the staying time has a weaker impact on the final stationary distribution. If u is small, then the opposite is true.

4.4.3 Algorithms

We still employ stationary distribution to decide the preference score of an item. According to [21, 32], the stationary distribution π of CMAP can be solved using the following equations:

$$\begin{cases} \pi_i = \frac{\frac{\widetilde{\pi}_i}{-q_{ii}}}{\sum_{j=1}^{S} \frac{\widetilde{\pi}_j}{-q_{jj}}}; \\ \widetilde{\pi}_j = \sum_{i \in S} \widetilde{\pi}_i \frac{q_{ij}}{-q_{ii}}. \end{cases} \tag{25}$$

The details of the algorithm are shown in Algorithm 1.

The main computation of our algorithm comes from two aspects: 1) probability transition matrix building; and 2) stationary distribution calculation. The main part for the first calculation is the similarity of current user to other users in the training set. In both KRCC and cosine similarities defined in our model, the complexity is $O(n)$, where n is the number of common items between the users. For the second aspect, we can conclude from Eq. (25) that it is a linear function of the stationary distribution of DMP. Thus the complexity is approximately the same with the one of DMP's stationary distribution calculation, which is $O(m)$ (m is the number of items) by utilizing the iterative power method. Therefore, the computation complexity scales linearly with respect to the number of items and users, indicating that our algorithm can be applied to very large datasets. In our experiments, the testing hardware environment is on two Windows workstations with four dual-core 2.5GHz CPU and 8GB physical memory each. The approximate total time for calculation in Netflix dataset is around 7 hours.

Table 1: Statistics of MovieLens and Netflix

Statistics	MovieLens	Netflix
Avg. Num. of Ratings/User	106.04	209.25
Avg. Num. of Ratings/Item	59.45	5654.50
Min. Num. of Ratings/User	20	1
Min. Num. of Ratings/Item	1	3
Max. Num. of Ratings/User	737	17653
Max. Num. of Ratings/Item	583	232944
Density of User/Item Matrix	6.3%	1.18%

5. EXPERIMENTS

In this section, we will first introduce the datasets. The experiments are conducted for three parts. The first part is an empirical study of quality-based and relevance-based algorithms, which serves as a quantitative analysis for the relationship of the two criteria in recommendations. The second part is to evaluate the recommendation performance of our proposed framework. The third part is to do the sensitivity analysis of CMAP.

5.1 Datasets

In this paper, we choose two datasets, MovieLens[2] and Netflix for experimental verification. In MovieLens, there are 100,000 ratings for 1,682 movies from 943 users. In Netflix, the size is much larger. It contains about 100,000,000 ratings from over 480,000 users for 17,770 movies. In both datasets, ratings are given as an integer value on the scale of 1 to 5, with higher value indicating better satisfaction. More statistics are shown in Table 1. In MovieLens, referring to the experimental setup in [22, 38], we randomly choose 600 users for training and the remaining 343 users for testing. In Netflix, we randomly divide the users into 10 groups. In each group, 80% users are randomly selected as training and the remaining 20% for testing. The average is calculated as the final result. To observe the performances when the active users have different number of ratings as history, experiments are conducted by selecting 5, 10 and 15 ratings as rating history for each active user respectively in MovieLens and 5, 10, and 20 in Netflix. We name them Given5, Given10, Given15, and Given20. Users whose rating number is less than the configuration are not included in evaluations. Before experiments, a pre-processing is conducted to rank all the ratings of a user in ascent order according to the rating time stamp.

5.2 Quantitative Analysis of Quality and Relevance

In this section, quantitative analysis of competitive quality-based and relevance-based algorithms on multiple criteria is conducted.

[2]http://www.cs.umn.edu/Research/GroupLens

Table 2: Performance on quality-based NDCG

Methods	Given5			Given10			Given15		
	NDCG1	NDCG3	NDCG5	NDCG1	NDCG3	NDCG5	NDCG1	NDCG3	NDCG5
PMF	0.635	0.612	0.623	0.644	0.646	0.654	0.696	0.689	0.698
EigenRank	**0.698**	**0.685**	**0.679**	**0.699**	**0.696**	**0.698**	**0.713**	**0.707**	**0.719**
Assoc	0.529	0.542	0.560	0.597	0.593	0.595	0.615	0.610	0.627
Freq	0.642	0.600	0.596	0.636	0.607	0.610	0.638	0.618	0.632

Table 3: Performance on relevance-based NDCG

Methods	Given5			Given10			Given15		
	NDCG1	NDCG3	NDCG5	NDCG1	NDCG3	NDCG5	NDCG1	NDCG3	NDCG5
PMF	0.333	0.325	0.309	0.241	0.227	0.212	0.198	0.194	0.186
EigenRank	0.326	0.306	0.304	0.279	0.282	0.285	0.274	0.276	0.275
Assoc	0.518	0.484	0.467	0.466	**0.459**	**0.449**	**0.455**	**0.426**	**0.430**
Freq	**0.539**	**0.489**	**0.477**	**0.478**	0.429	0.412	0.428	0.377	0.364

The purpose is to evaluate quality-based algorithms' performances on relevance-based metric and relevance-based algorithms' performances on quality-based metric. In the experiments, two quality-based algorithms and two relevance-based algorithms are chosen. Quality-based algorithms include PMF [28] and EigenRank [20]. Relevance-based algorithms include the association-based method (Assoc) [7] and a hitting-frequency-based method (Freq) [33]. The experiments are conducted on both MovieLens and Netflix. We only report the results for MovieLens in Table 2 and Table 3 due to space limitation. Similar results can be observed from Netflix. From the experimental results, we can conclude that both quality-based and relevance-based methods do not perform well in the opposite metric. For configuration of "Given5,NDCG1", quality-based algorithms outperform relevance-based algorithms by 8.7% in quality-based NDCG and relevance-based algorithms outperform quality-based algorithms by 65.3% in relevance-based NDCG. This can quantitatively support the importance of fusing quality-based and relevance-based algorithms together in recommender systems. In quality-based NDCG metric, EigenRank outperforms PMF in almost all the configurations which also supports the reason to choose EigenRank as a fundamental quality-based algorithm to combine.

5.3 Recommendation Performance

The experiments conducted for overall performance aim at the following three issues: 1) Quantitatively, how about the performances of the three combination methods comparing to competitive quality-based and relevance-based algorithms? 2) Qualitatively, whether the incompleteness problem in each single-criteria approach can be solved by our framework? 3) How can CMAP outperform traditional combination methods? For Issue 1, we compare results of the three combination methods with the quality-based and relevance-based baselines; for Issue 2, we make statistics of our CMAP framework on the quality-relevance balance study discussed before; and for Issue 3, we make comparisons among the three combination methods.

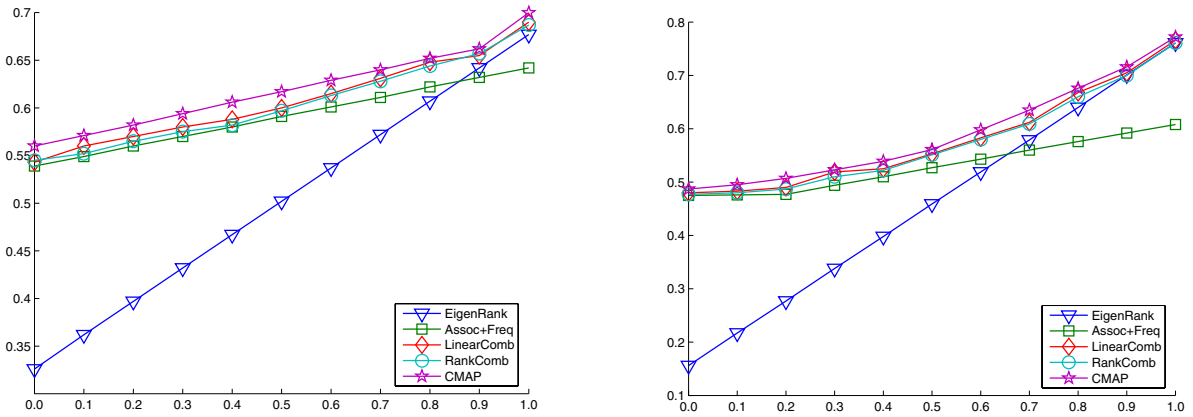
Within quality-based methods, according to previous study, we choose EigenRank [20] as baseline method, because it outperforms PMF in almost all the configurations in both quality-based and relevance-based NDCG metrics. Within relevance-based methods, as the assoication-based method [7] and the hitting-frequency-based method [33] have their own advantages in different cases as shown before, we choose the best result from them as our baseline method.

Fig. 4(a) shows the overall performance on MovieLens; and Fig. 4(b) shows the overall performance on Netflix. In these two figures, the experimental configuration is: Given5, NDCG1. In other configurations, similar results can be obtained (See Table 4 and Table 5.). In both these two figures, we can observe that the three combination methods outperform the two single-criteria methods in almost all the settings of λ. If we average results from $\lambda = 0.6$ to $\lambda = 1$ as quality-bias metric, and average results from $\lambda = 0$ to $\lambda = 0.4$ as relevance-bias metric. In quality-bias metric, the combination method outperforms quality-based algorithm by 8.2% in MovieLens and 6.2% in Netflix; in relevance-bias metric, our approach outperforms relevance-based algorithm by 4.1% in MovieLens and 4.9% in Netflix.

Fig. 5 shows the distribution of recommended results of the top five items by CMAP. The parameters are adapted for $\lambda = 0.8$ (quality-bias) in the left, and $\lambda = 0.2$ (relevance-bias) in the right. It can be obtained that in both figures, there is a quantity of items for both Type A and D, indicating that the framework is effective in solving the incompleteness limitation of single-criteria methods. In addition, in quality-bias CMAP, recommended items are likely to have high ratings; and in relevance-bias CMAP, they are likely to have high hitting count. This indicates that the recommended results of CMAP can adapt for different balance requests from users in practical applications.

Among the three combination methods, CMAP performs the best in almost all the settings of λ. In average of all the λ configuration, for Given5 and NDCG1, the CMAP model outperforms LinearComb by 2.0% in MovieLens and 2.0% in Netflix; and it also outperforms RankComb by 3.0% in MovieLens and 2.7% in Netflix. RankComb performs the worst, because it misses quantity information. The advantage of our approach comparing to LinearComb is that the latter unnaturally combines probability and count value linearly, which are incompatible scores; while in CMAP, the combination has an practical interpretation explained before.

At first, we expect that the accuracy of relevance-based algorithm will decrease when λ increases, which is not true according to experimental results. There are two reasons. 1) Although quality and relevance are emphasizing different aspects, there is some correlative relation between them. In these two datasets, if an item is relevant to a user, it will have great chance to have good quality; but the opposite is not true that many high-quality items do not attract that many users. 2) In experiments, we approximately utilize the rating record as visited record. In fact, it is more practical to use the real visited record. Because many users will not take time to rate an item after they visit them. Thus our experiments will cause some bias to quality.



(a) MovieLens (b) Netflix

Figure 4: Recommendation performance

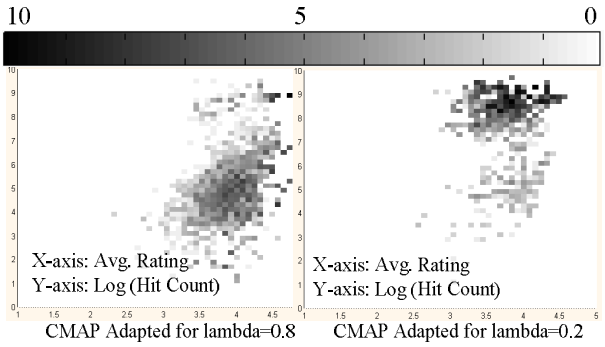


Figure 5: Distribution of recommended results of CMAP

5.4 Sensitivity Analysis

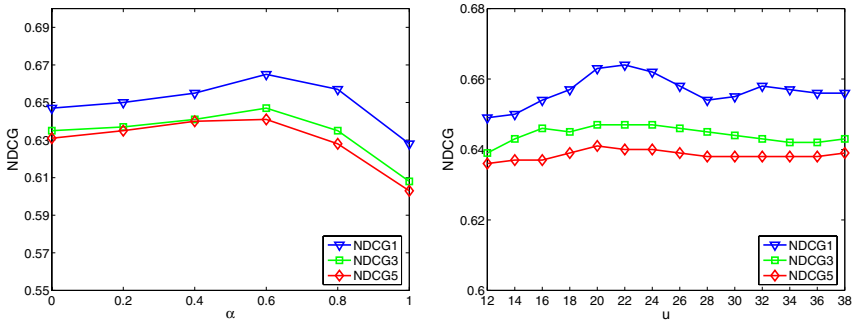
There are two important parameters in our approach: α in Eq. (15) and u in Eq. (24). α balances CMAP's probability transition matrix between rating preference order and association feature. It scales from 0 to 1. When $\alpha = 1$, the transition matrix is built from quality-based information only; when $\alpha = 0$, it is built from relevance-based information only; in other cases, it is a fusion of two kinds of information. u is the service rate at ticket windows which controls the influence of staying time of states. As discussed before, when u is small, the staying time will have greater impact on the stationary distribution; and when u is large, the transition probability will have greater impact. Fig. 6(a) shows the impact of α on MovieLens, given 10 ratings as history, with $\lambda = 0.8$ and $u = 20$ and Fig. 7(a) shows the impact of α on Netflix, given 10 ratings as history, with $\lambda = 0.2$ and $u = 40$. This is a general example, and similar results can be obtained in other configurations in both Netflix and MovieLens. Fig. 6(b) shows the impact of u on MovieLens, given 10 ratings as history, with $\lambda = 0.8$ and $\alpha = 0.6$. Fig. 7(b) shows the impact of u on Netflix, given 10 ratings as history, with $\lambda = 0.2$ and $\alpha = 0.4$.

6. RELATED WORK

Recommendation has been treated as a classical regression problem to predict ratings over the decades. Collaborative filtering methods are widely employed for this problem with two categories, neighborhood models and factorization models. The idea of neighborhood methods is to predict ratings from similar users and items. Thus approaches are naturally divided into user-based [4, 14] and item-based [19, 29], together with some combined approaches [35]. The short coming is that when no similar neighbor exits, it is hard to get accurate prediction. Alternatively, factorization methods [16, 28, 31, 23] are from a whole perspective of all the users and items. Usually, it build a latent feature space for all the items and users from training data. Thus to some extent, it will solve the disadvantages of neighborhood methods. But it cannot utilize the strong association pattern among similar neighbors, which is just the advantage of neighborhood methods. There are also some ensemble methods that combine neighborhood methods and factorization methods. [26, 38, 37] combine the results of them together and [17] propose to build an unified model, which has been demonstrated the most effective in the Netflix competition. Other recent competitive methods include [20, 28, 36, 40, 41].

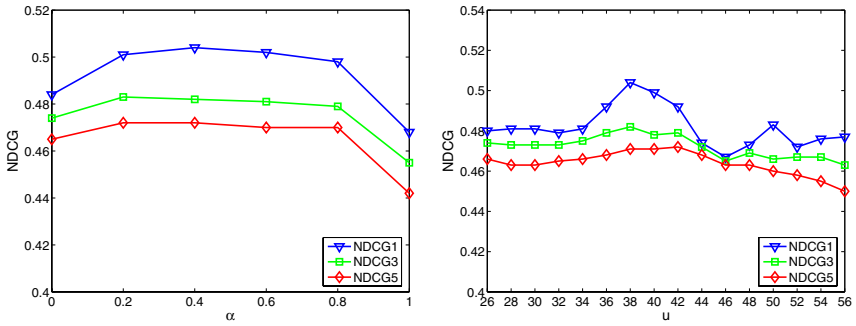
Apart from traditional recommendation in the criterion of quality, there are some work emphasize on more other criteria, include relevance [7], coverage [34], diversity [11], etc. Relevance-based recommendations mainly depend on association feature [7, 30] and hitting frequency feature [3, 18, 33]. The basic assumption of the former is that frequent co-occurrence items in the past are also likely to appear together in the future. Thus a statistical analysis is made on each item pair, and the recommendation results are based on the co-occurrence frequency. An intuitive interpretation of the latter is that popular items are likely to interest users. In "Who Rate What" task of KDD-cup 2007, the weight of this feature is much larger than others [18].

Metrics for recommender systems contain both quality-based and relevance-based. Quality-based metrics measure the closeness of a recommender system's predicted ratings to the users' real ratings. Classical metric is Mean Absolute Error (MAE), which has been widely used in previous work [4, 16, 29]. Some variances of MAE include mean squared error (MSE), root mean squared error (RMSE), etc. Relevance-based metrics, on the other hand, measure the likelihood that an item will be hit. Classical metrics used in previous work include precision/recall [7], Receiver Operating Characteristic (ROC) curve, etc. In application systems, a rank list is the final output for users. Thus normalized discount cumulated



(a) Impact of α (b) Impact of u

Figure 6: Impact of parameters of CMAP in MovieLens



(a) Impact of α (b) Impact of u

Figure 7: Impact of parameters of CMAP in Netflix

gain (NDCG), a metric to evaluate rank problem [6], has been recently employed to evaluate recommendation results. These metrics include both quality-based NDCG [20] and relevance-based NDCG [13] in recommendations.

7. CONCLUSIONS

In this paper, we make a preliminary work on multi-criteria recommendation. We take quality and relevance as two criteria in recommender system for analysis. We study the interplay relationship of they impact each other and show that both quality-based and relevance-based methods cannot perform well in the opposite criteria. As the first ever solutions, we propose an integrated metric considering both criteria. Then we investigate how to combine previous work to adjust the new metric under the concept of multi-criteria recommendation. We propose a CMAP framework that enables principled and natural integration with features derived from both quality-based and relevance-based algorithms. Through empirical study on two real world datasets, we demonstrate that the combined approach can significantly outperform traditional quality-based and relevance-based algorithms. The framework has linear computational complexity. Thus from a technical standpoint, we believe the work in this paper will be helpful in improving recommender system in real applications.

8. ACKNOWLEDGMENTS

The work described in this paper was fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK4154/10E and No. CUHK4152/10E), and a Google Focused Grant Project under "Mobile 2014". The authors also would like to thank the reviewers for their helpful comments.

9. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE TKDE*, 17(6):734–749, 2005.

[2] W. Anderson. *Continuous-time Markov chains: Applications-oriented Approach*. Springer, 1991.

[3] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Proc. of the NCAI*, pages 714–720, 1998.

[4] J. Breese, D. Heckerman, C. Kadie, et al. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of UAI 98*, pages 43–52, 1998.

[5] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[6] B. Croft, D. Metzler, and T. Strohman. *Search Engines: Information Retrieval in Practice*. Addison Wesley, 2009.

[7] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[8] M. Dummett. The Borda count and agenda manipulation. *Social Choice and Welfare*, 15(2):289–296, 1998.

[9] A. Elberse. Should you invest in the long tail? *harvard business review*, 86(7/8):88–96, 2008.

[10] A. Elberse and F. Oberholzer-Gee. Superstars and underdogs: an examination of the long tail phenomenon in video sales. *MSI Reports*, 45(4):49–72, 2007.

[11] D. Fleder and K. Hosanagar. Recommender systems and their impact on sales diversity. In *Proc. of EC 2007*, page 199. ACM, 2007.

[12] D. Frank Hsu and I. Taksa. Comparing rank and score combination methods for data fusion in information retrieval. *Information Retrieval*, 8(3):449–480, 2005.

[13] Z. Guan, J. Bu, Q. Mei, C. Chen, and C. Wang. Personalized tag recommendation using graph-based ranking on multi-type interrelated objects. In *Proc. of SIGIR'09*, pages 540–547, 2009.

[14] J. Herlocker, J. Konstan, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proc. of SIGIR 99*, pages 230–237. ACM New York, NY, USA, 1999.

Table 4: Overall performance for other settings on MovieLens

α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Given5,NDCG3											
EigenRank	0.306	0.344	0.381	0.419	0.456	0.493	0.531	0.568	0.605	0.643	0.680
Assoc+Freq	0.489	0.500	0.511	0.522	0.533	0.544	0.555	0.567	0.578	0.589	0.600
LinearComb	0.502	0.511	0.528	0.538	0.559	0.568	0.587	0.598	0.617	0.648	0.678
RankComb	0.503	0.516	0.524	0.533	0.552	0.567	0.583	0.596	0.613	0.637	0.681
CMAP	0.514	0.525	0.541	0.553	0.566	0.580	0.596	0.607	0.624	0.651	0.685
Given5,NDCG5											
EigenRank	0.304	0.341	0.379	0.416	0.454	0.491	0.529	0.566	0.604	0.641	0.679
Assoc+Freq	0.477	0.489	0.501	0.513	0.524	0.537	0.548	0.560	0.572	0.584	0.596
LinearComb	0.484	0.498	0.513	0.531	0.546	0.561	0.576	0.593	0.608	0.645	0.682
RankComb	0.487	0.500	0.511	0.524	0.537	0.554	0.571	0.588	0.606	0.639	0.681
CMAP	0.499	0.511	0.523	0.538	0.556	0.569	0.585	0.598	0.623	0.655	0.685
Given10,NDCG1											
EigenRank	0.279	0.321	0.363	0.404	0.446	0.487	0.528	0.570	0.611	0.653	0.694
Assoc+Freq	0.478	0.493	0.500	0.525	0.541	0.557	0.572	0.588	0.604	0.620	0.636
LinearComb	0.510	0.527	0.540	0.560	0.581	0.600	0.620	0.640	0.660	0.680	0.710
RankComb	0.479	0.495	0.516	0.536	0.556	0.578	0.607	0.633	0.659	0.683	0.701
CMAP	0.526	0.544	0.559	0.578	0.597	0.611	0.628	0.648	0.664	0.683	0.716
Given10,NDCG3											
EigenRank	0.282	0.323	0.364	0.405	0.446	0.487	0.527	0.568	0.609	0.650	0.691
Assoc+Freq	0.459	0.472	0.485	0.499	0.513	0.526	0.539	0.553	0.571	0.589	0.607
LinearComb	0.457	0.479	0.500	0.519	0.538	0.568	0.589	0.608	0.629	0.659	0.700
RankComb	0.462	0.483	0.500	0.519	0.528	0.557	0.584	0.608	0.636	0.661	0.703
CMAP	0.465	0.488	0.509	0.532	0.555	0.576	0.598	0.620	0.643	0.666	0.705
Given10,NDCG5											
EigenRank	0.285	0.326	0.366	0.407	0.447	0.488	0.528	0.569	0.609	0.650	0.690
Assoc+Freq	0.449	0.463	0.478	0.493	0.507	0.522	0.536	0.551	0.570	0.590	0.610
LinearComb	0.439	0.459	0.478	0.509	0.527	0.558	0.579	0.608	0.629	0.648	0.701
RankComb	0.452	0.470	0.491	0.512	0.523	0.549	0.574	0.608	0.635	0.656	0.703
CMAP	0.449	0.473	0.496	0.519	0.543	0.565	0.589	0.613	0.637	0.659	0.703
Given15,NDCG1											
EigenRank	0.274	0.316	0.358	0.400	0.443	0.485	0.527	0.569	0.611	0.653	0.695
Assoc+Freq	0.455	0.471	0.487	0.503	0.519	0.535	0.554	0.575	0.596	0.617	0.638
LinearComb	0.480	0.501	0.522	0.544	0.564	0.586	0.612	0.631	0.652	0.673	0.711
RankComb	0.456	0.478	0.492	0.513	0.522	0.551	0.584	0.614	0.641	0.673	0.706
CMAP	0.498	0.517	0.539	0.558	0.578	0.597	0.616	0.639	0.658	0.678	0.722
Given15,NDCG3											
EigenRank	0.276	0.319	0.362	0.405	0.447	0.490	0.532	0.575	0.617	0.660	0.702
Assoc+Freq	0.426	0.445	0.463	0.482	0.500	0.518	0.537	0.554	0.573	0.594	0.618
LinearComb	0.436	0.462	0.486	0.511	0.529	0.558	0.588	0.611	0.638	0.659	0.711
RankComb	0.433	0.454	0.472	0.498	0.513	0.544	0.575	0.609	0.638	0.668	0.711
CMAP	0.453	0.474	0.494	0.515	0.538	0.564	0.590	0.615	0.645	0.669	0.719
Given15,NDCG5											
EigenRank	0.275	0.319	0.363	0.407	0.451	0.495	0.539	0.583	0.626	0.670	0.714
Assoc+Freq	0.430	0.450	0.470	0.489	0.509	0.529	0.548	0.568	0.588	0.607	0.632
LinearComb	0.429	0.448	0.469	0.500	0.528	0.558	0.579	0.609	0.637	0.666	0.715
RankComb	0.430	0.451	0.472	0.497	0.518	0.549	0.583	0.615	0.646	0.674	0.720
CMAP	0.434	0.459	0.485	0.514	0.538	0.568	0.592	0.623	0.649	0.679	0.727

Table 5: Overall performance on other settings on Netflix

α	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Given5,NDCG3											
EigenRank	0.155	0.215	0.275	0.335	0.395	0.456	0.516	0.576	0.636	0.696	0.756
Assoc+Freq	0.456	0.463	0.469	0.486	0.504	0.523	0.541	0.560	0.578	0.597	0.615
LinearComb	0.455	0.464	0.472	0.493	0.521	0.551	0.575	0.600	0.649	0.701	0.763
RankComb	0.456	0.464	0.471	0.492	0.522	0.543	0.575	0.601	0.648	0.691	0.756
CMAP	0.459	0.469	0.487	0.506	0.527	0.559	0.593	0.624	0.658	0.706	0.766
Given5,NDCG5											
EigenRank	0.173	0.231	0.289	0.350	0.405	0.463	0.520	0.578	0.636	0.694	0.752
Assoc+Freq	0.420	0.450	0.460	0.480	0.500	0.520	0.540	0.560	0.580	0.600	0.620
LinearComb	0.421	0.454	0.471	0.487	0.516	0.551	0.585	0.618	0.650	0.700	0.759
RankComb	0.421	0.450	0.464	0.485	0.518	0.552	0.583	0.617	0.648	0.698	0.755
CMAP	0.440	0.457	0.478	0.498	0.523	0.554	0.588	0.624	0.659	0.705	0.765
Given10,NDCG1											
EigenRank	0.163	0.224	0.285	0.346	0.407	0.469	0.530	0.591	0.652	0.713	0.774
Assoc+Freq	0.452	0.456	0.466	0.485	0.505	0.525	0.545	0.565	0.584	0.604	0.624
LinearComb	0.460	0.464	0.482	0.502	0.531	0.560	0.588	0.626	0.669	0.719	0.778
RankComb	0.460	0.463	0.473	0.498	0.527	0.558	0.588	0.625	0.665	0.718	0.777
CMAP	0.477	0.494	0.509	0.528	0.544	0.574	0.607	0.643	0.678	0.725	0.785
Given10,NDCG3											
EigenRank	0.163	0.224	0.284	0.345	0.405	0.466	0.526	0.587	0.647	0.707	0.768
Assoc+Freq	0.439	0.448	0.461	0.482	0.503	0.524	0.544	0.565	0.586	0.607	0.628
LinearComb	0.441	0.456	0.470	0.501	0.532	0.559	0.588	0.619	0.660	0.709	0.771
RankComb	0.442	0.455	0.469	0.498	0.530	0.559	0.584	0.613	0.654	0.705	0.770
CMAP	0.446	0.468	0.487	0.505	0.535	0.568	0.599	0.632	0.666	0.717	0.778
Given10,NDCG5											
EigenRank	0.184	0.241	0.298	0.355	0.412	0.469	0.525	0.582	0.639	0.696	0.753
Assoc+Freq	0.408	0.420	0.455	0.478	0.500	0.523	0.545	0.568	0.590	0.612	0.635
LinearComb	0.426	0.439	0.458	0.491	0.525	0.547	0.581	0.625	0.660	0.703	0.764
RankComb	0.426	0.448	0.456	0.489	0.522	0.548	0.578	0.624	0.658	0.703	0.763
CMAP	0.433	0.455	0.477	0.497	0.527	0.558	0.599	0.634	0.668	0.716	0.775
Given20,NDCG1											
EigenRank	0.147	0.210	0.272	0.335	0.398	0.461	0.523	0.586	0.648	0.711	0.774
Assoc+Freq	0.430	0.450	0.470	0.490	0.510	0.530	0.550	0.570	0.590	0.610	0.630
LinearComb	0.448	0.462	0.475	0.506	0.536	0.566	0.597	0.627	0.668	0.719	0.773
RankComb	0.442	0.458	0.472	0.502	0.532	0.562	0.593	0.621	0.666	0.718	0.773
CMAP	0.466	0.478	0.492	0.517	0.546	0.577	0.612	0.649	0.688	0.731	0.774
Given20,NDCG3											
EigenRank	0.154	0.216	0.277	0.339	0.400	0.462	0.524	0.585	0.647	0.708	0.770
Assoc+Freq	0.423	0.437	0.459	0.481	0.503	0.526	0.548	0.570	0.592	0.614	0.636
LinearComb	0.425	0.447	0.468	0.500	0.522	0.555	0.588	0.621	0.665	0.709	0.780
RankComb	0.424	0.439	0.469	0.499	0.525	0.556	0.587	0.619	0.663	0.711	0.780
CMAP	0.433	0.455	0.479	0.504	0.537	0.569	0.604	0.637	0.671	0.718	0.780
Given20,NDCG5											
EigenRank	0.167	0.226	0.285	0.343	0.402	0.461	0.519	0.579	0.637	0.696	0.755
Assoc+Freq	0.415	0.429	0.453	0.477	0.500	0.524	0.548	0.571	0.595	0.618	0.642
LinearComb	0.420	0.438	0.456	0.490	0.514	0.550	0.583	0.620	0.665	0.712	0.777
RankComb	0.420	0.435	0.457	0.489	0.513	0.547	0.581	0.618	0.663	0.713	0.776
CMAP	0.424	0.447	0.472	0.498	0.527	0.563	0.598	0.635	0.671	0.715	0.780

[15] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[16] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22(1):89–115, 2004.

[17] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proc. of SIGKDD 08*, pages 426–434. ACM, 2008.

[18] M. Kurucz, A. Benczúr, T. Kiss, I. Nagy, A. Szabó, and B. Torma. Who Rated What: a combination of SVD, correlation and frequent sequence mining. In *Proc. KDD Cup and Workshop*, volume 23, pages 720–727, 2007.

[19] G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 2003.

[20] N. Liu and Q. Yang. EigenRank: a ranking-oriented approach to collaborative filtering. In *Proc. of SIGIR 08*, pages 83–90. ACM New York, NY, USA, 2008.

[21] Y. Liu, B. Gao, T. Liu, Y. Zhang, Z. Ma, S. He, and H. Li. BrowseRank: letting web users vote for page importance. In *Proc. of SIGIR 08*, pages 451–458. ACM New York, NY, USA, 2008.

[22] H. Ma, I. King, and M. Lyu. Effective missing data prediction for collaborative filtering. In *Proc. of SIGIR 07*, pages 39–46. ACM New York, NY, USA, 2007.

[23] H. Ma, I. King, and M. Lyu. Learning to recommend with social trust ensemble. In *Proc. of ACM SIGIR'09*, pages 203–210. ACM, 2009.

[24] J. Marden. *Analyzing and modeling rank data*. Chapman & Hall/CRC, 1995.

[25] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. 1998.

[26] D. Pennock, E. Horvitz, S. Lawrence, and C. Giles. Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach. In *Proceedings of UAI 00*, pages 473–480. Stanford, California, 2000.

[27] S. Rosset, C. Perlich, and Y. Liu. Making the most of your data: KDD Cup 2007" How Many Ratings" winner's report. 2007.

[28] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. *Advances in neural information processing systems*, 20, 2008.

[29] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *Proc. of WWW 01*, pages 285–295. ACM New York, NY, USA, 2001.

[30] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *Proc. of EC 00*, pages 158–167, New York, NY, USA, 2000. ACM.

[31] L. Si and R. Jin. Flexible mixture model for collaborative filtering. In *Proc. of ICML'03*. ACM, NY, USA, 2003.

[32] W. Stewart. *Numerical solution of Markov chains*. CRC Press, 1991.

[33] J. Sueiras, A. Salafranca, and J. Florez. A classical predictive modeling approach for Task "Who rated what?" of the KDD CUP 2007. *Proceedings of the KDD Cup*, page 34, 2007.

[34] E. Vozalis and K. Margaritis. Analysis of recommender systems algorithms. In *Proc. of HERCMA-2003*, 2003.

[35] J. Wang, A. De Vries, and M. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proc. of SIGIR 2006*, pages 501–508. ACM New York, NY, USA, 2006.

[36] M. Weimer, A. Karatzoglou, Q. Le, and A. Smola. Maximum Margin Matrix Factorization for Collaborative Ranking. *Advances in neural information processing systems*, 2007.

[37] X. Xin, I. King, H. Deng, and M. Lyu. A social recommendation framework based on multi-scale continuous conditional random fields. In *Proc. of CIKM'09*, pages 1247–1256. ACM, 2009.

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

[39] K. Yu, S. Zhu, J. Lafferty, and Y. Gong. Fast nonparametric matrix factorization for large-scale collaborative filtering. In *Proc. of SIGIR'09*, pages 211–218. ACM, NY, USA, 2009.

[40] Y. Zhang and J. Koren. Efficient bayesian hierarchical user modeling for recommendation system. In *Proc. of SIGIR 2007*, pages 47–54. ACM, 2007.

[41] S. Zhu, K. Yu, and Y. Gong. Stochastic relational models for large-scale dyadic data using MCMC. *Advances in Neural Information Processing Systems*, 21:1993–2000, 2009.