Effective Fusion-based Approaches for Recommender Systems

XIN, Xin

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
July 2011

Thesis / Assessment Committee Members

Professor Wai LAM (Chair)
Professor Irwin Kuo Chin KING (Thesis Supervisor)
Professor Michael Rung Tsong LYU (Thesis Supervisor)
Professor John Chi Shing LUI (Committee Member)
Professor Qiang YANG (External Examiner)

Abstract of thesis entitled:

Effective Fusion-based Approaches for Recommender Systems

Submitted by XIN, Xin

for the degree of Doctor of Philosophy

at The Chinese University of Hong Kong in July 2011

Recommender systems are important nowadays. With the explosive growth of resources on the Web, users encounter information overload problem. The research issue of recommender systems is a kind of information filtering technique that suggests user-interested items (e.g., movies, books, products, etc.) to solve this problem. Collaborative filtering (CF) is the key approach. Over the decades, recommender systems have been demonstrated important in E-business. Thus designing accurate algorithms for recommender systems has attracted much attention.

This thesis is to investigate effective fusion-based approaches for recommender systems. Effective fusion of various features and algorithms becomes important along with the development of recommendation techniques. Because each feature/algorithm has its own advantages and disadvantages. A combination to get the best performance is desired in applications. The fusion-based approaches investigated are from the following four levels.

(1) Relational fusion of multiple features for the classical regression task (single measure and dimension). Originally, the task of recommender systems is formulated as a regression task. Many CF algorithms and fusion methods have been proposed. The limitation of previous fusion methods is that only local

features are utilized and the global relational dependency is ignored, which would impair the performance of CF. We propose a relational fusion approach based on conditional random fields (CRF) to improve traditional fusion methods by incorporating global relational dependency.

(2) Fusion of regression-oriented and ranking-oriented algorithms for multi-measure adaption. Beyond the level of classical regression, ranking the items directly is another important task for recommender systems. A good algorithm should adapt to both regression-oriented and ranking-oriented measures. Traditionally, algorithms separately adapt to a single one, thus they cannot adapt to the other. We propose methods to combine them to improve the performances in both measures.

(3) Fusion of quality-based and relevance-based algorithms for multi-dimensional adaption. Recommender systems should consider the performances of multiple dimensions, such as quality and relevance. Traditional algorithms, however, only recommend either high-quality or high-relevance items. But they cannot adapt to the other dimension. We propose both fusion metrics and fusion approaches to effectively combine multiple dimensions for better performance in multi-dimensional recommendations.

(4) Investigation of impression efficiency optimization in recommendation. Besides performance, impression efficiency, which describes how much profit can be obtained per impression of recommendation, is also a very important issue. From recent study, over-quantity recommendation impression is intrusive to users. Thus the impression efficiency should be formulated and optimized. But this issue has rarely been investigated. We formulate the issue under the classical secretary problem framework and extend an online secretary algorithm to solve it.

論文題目：推薦系統的有效融合方法

作者：辛欣

學校：香港中文大學

學系：電腦科學與工程學系

修讀學位：哲學博士

摘要：

推薦系統是當今重要的研究課題。隨著網路資料的指數增長，用戶面臨著資訊超載的問題。推薦系統的研究課題是研究一種能夠自動向用戶推薦專案（包括電影，書籍，產品等）的資訊過濾技術，從而解決這個問題。目前推薦系統的主要方法是基於協同過濾的方法。十多年來，推薦系統在電子商務中的重要性已被證實。設計準確的推薦系統已經爲廣泛研究人員所關注。

本文主要的研究內容是資訊系統中推薦技術的有效融合方法。隨著推薦技術的發展，有效的融合技術也變得越來越重要。因爲每一種特徵或方法都有其各自的優缺點，所以在應用中需要將它們融合起來使其互補。本文研究的融合技術主要基於如下四個層次。

(1) 針對傳統推薦問題（單評測單維度）的多特徵關係融合。傳統的推薦問題是一個回歸問題。人們提出了許多協同過濾的方法。以往融合方法的缺點是，這些方法只用了局部特徵，而全局的關係特徵卻被忽略，從而破壞了推薦系統的性能。我們提出了一種基於條件隨機場的關係融合方法，它能夠將全局的關係特徵考慮進來，從而提高推薦系統的性能。

(2) 面向回歸推薦演算法和面向排序推薦演算法的多評測融合。除了傳統的回歸問題，對專案進行直接排序是資訊系統的另一個重要任務。一個好的演算法應該同時具有良好的回歸表現和排序表現。而傳統的方法是單獨爲單評測而設計，從而不能同時適應多種評測方法。我們在本文提出將面向回歸演算法和面向排序演算法融合的方法，從而使推薦性能在多評測上都得到提高。

(3) 基於品質推薦演算法和基於興趣推薦演算法的多維度融合。推薦系統應該考慮其在多個維度的性能，比如品質維度，興趣維度等。而傳統的方法只考慮一種的一種。他們或者推薦品質好的專案，或者推薦用戶興趣高的專案。但是每一種演算法不能適應其他的維度。因此在本文中我們提出了多維度推薦系統的評測標準和有效的融合方法，使推薦系統在多維度的性能上得到提高。

(4) 推薦系統的顯示效率優化。除了推薦系統的性能，其顯示效率也是一個重要的課題。顯示效率是指向用戶顯示一條推薦結果推薦系統能夠獲得的平均收益。近年來的研究表明，過度的資訊推薦是對用戶的商業侵擾。因此資訊系統的顯示效率需要被優化。而對於這個問題的研究還比較少。本文對這個問題進行形式化的定義，並且在經典秘書問題的框架下提出有效的方法。

Acknowledgement

I would like to express my sincere appreciation to my supervisors, Prof. Irwin King and Prof. Michael R. Lyu. They give me strong support when I meet difficulties in the study. I am grateful to my thesis committee members, Prof. John C.S. Lui and Prof. Wai Lam for their helpful comments and suggestions about this thesis. I give my special thanks to Prof. Qiang Yang for being the external committee for this thesis.

I would like to thank my pervious supervisor, Prof. Juanzi Li, and previous mentor, Prof. Jie Tang, for their great help in my master study in Tsinghua University.

I would like to thank my dearest friends in Hong Kong, Wujie Zheng, Wei Yu, Kitty Liu, Yangfan Zhou, Wei Wang, Junjie Xiong, Fan Yang and Mingzhen Mo. I also thank my colleagues in the Machine Learning and Web Intelligence Group, Hao Ma, Hongbo Deng, Xinyu Chen, Xiaoqi Li, Chao Zhou, Shouyuan Chen, Haiqin Yang, Guang Ling and many others.

This work is dedicated to my parents.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The research of recommender systems is important nowadays. With the explosive growth of resources on the Web, users encounter information overload problem, which means that they are always facing against too much information on the Web and the useful information targeted is unfriendly covered by disgusting noises. The issue of recommender systems is a kind of information filtering technique to solve this problem. Specifically, the task of recommender systems is to suggest items on the Web (e.g., movies, books, products, etc.) to users according to their different tastes. Fig. 1.1 shows an example of recommender system in industry. It is Amazon[1], an E-business service. Users can register an account in the system for both selling and buying products online. The algorithm of recommender systems would suggest 5-10 products to each user among millions of products with hundreds of categories, based on his/her browsing and transaction history. For users, such kind of technique would be surely useful to make convenience of the Web usage. The benefits for E-business companies are also considerable. A good recommendation service has been demonstrated effective

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

Figure 1.1: Recommender system example in Amazon

to increase both user and transaction amount, which makes more profit. Over the decades, the importance and success of recommender systems have been demonstrated in both academia and industry.

The evaluation structure of recommender systems can be summarized as shown in Fig. 1.2.

- The bottom level: Single measure and single dimension adaption. The most popular and original evaluation for recommender systems is the regression-based measure for the dimension of quality. Usually, the quality of an item for a specific user can be presented by a rating, for example, an integer from 1 to 5 with the higher value for better quality. The task is to predict the ratings as accurate as possible to the user's real ratings. The main technique for recommender systems is collaborative filtering (CF). The idea is based on the assumption that similar users would

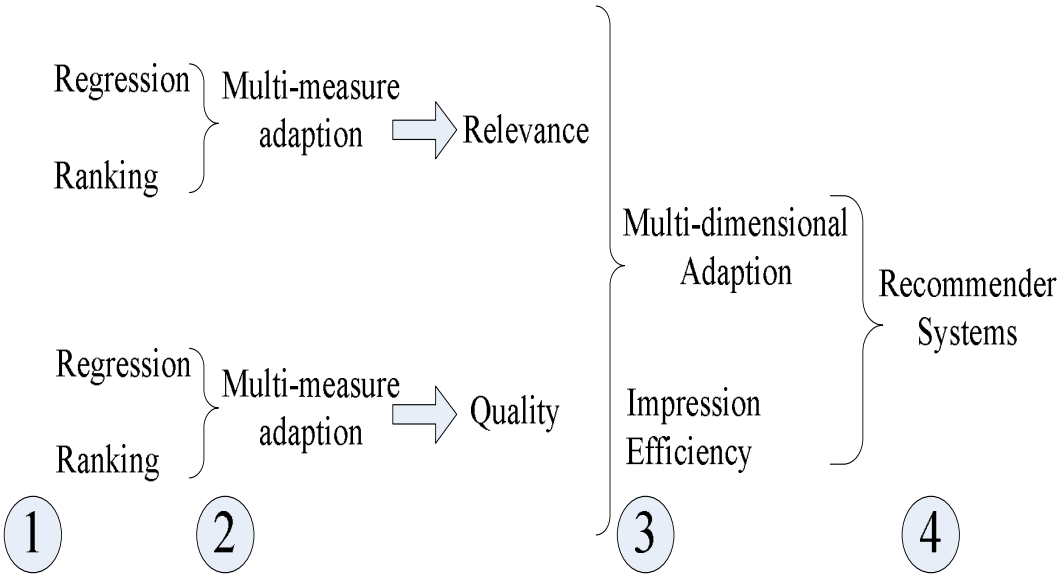


Figure 1.2: Evaluation structure of recommender systems

have similar tastes. Traditionally, there are two categories of CF methods: memory-based [17, 64, 130] methods and model-based [67, 152] methods.

- The second bottom level: Multi-measure adaption for recommender systems. Besides the performance of regression, ranking is also an important evaluation measure for recommender systems, proposed by recent work [91]. This second level means that recommender systems should adapt to both regression and ranking measures in their performances. Ranking-adapted algorithms emphasize on modeling the preference order of items for a user directly, and the rating predictions are not necessary. Thus it is different from regression task. By utilizing different loss functions and similarity calculations, collaborative filtering methods can be converted to adapt to ranking-oriented measures.

- The third level: Multi-dimensional adaption for recommender systems. The performance of recommender sys-

tems should adapt to multiple dimensions. Besides the dimension of quality in most traditional work, relevance is also an important dimension in the evaluation of recommender system [36]. Other typical dimensions also include coverage, diversity, etc. Traditional work in recommender systems is mainly focusing on recommending high-quality items. While other dimensions become more popular in recent work [36, 46, 80, 127, 149, 154]. All these dimensions should be considered for the successful performance of recommender systems.

- The last level: Successful performance with high impression efficiency. Besides the successful performance of recommender systems, impression efficiency should also be carefully considered. Impression efficiency describes how much profit can be obtained per impression of recommendation result. The reason why the impression efficiency should be optimized is that over-quantity impression of recommendation results would have commercial intrusion to users, which is identified deeply in recent work [18, 21, 33, 71, 99]. Currently, the impression efficiency of recommender systems is simply controlled by heuristic rules.

In this thesis, we investigate effective fusion-based approaches for recommender systems. Over the decades, a number of features and algorithms are proposed. But each has its own advantages and disadvantages. Therefore, effective fusion-based techniques to get the best performance among these algorithms are desired in applications. The goal of the thesis is to investigate effective fusion techniques at the four different levels as discussed above, in order to solve the following four limitations.

1. **Limitation at level 1: The relational dependency is ignored in previous fusion methods for single measure and dimension.** Although many fusion methods

have been proposed to fuse various features, their predictions of user-item pairs are independent with each other. This means that only local features are utilized, while the global relational dependency is ignored. But the ratings of user-item pairs are correlated with each other. For example, similar items are assumed to have similar ratings. If only local features are considered, the predictions would depend on the observed user-item pairs only. Thus many user-item pairs for prediction would fail to find reliable information under the sparse data environment of recommender systems. If relational features are considered, all the predictions would depend on each other besides the observed user-item pairs. Thus the information is richer, which would improve recommendation performance, especially when the data is sparse.

2. **Limitation at level 2: The fusion of multi-measure adaption is rarely investigated, and single-measure-adapted algorithms cannot adapt to multi-measure performance.** Ideally, recommender systems should have good performances in both regression and ranking measures. However, most current studies treat them as two separate tasks. The problem is that they may fail to adapt to each other. As a result, the performance is limited. Thus the separated tasks are expected to be combined together.

3. **Limitation at level 3: The fusion of multi-dimensional adaption is rarely investigated, and single-dimension-adapted algorithms cannot adapt to multi-dimensional performance.** Successful recommender systems should simultaneously consider multi-dimensional performance. Previous work, however, is mainly focusing on a single dimension. The limitation is that the single-dimensional recommendation results may not adapt to other dimensions in

many cases. Thus the correlations among different dimensions should be studied and the combination algorithms are expected to be explored.

4. **Limitation at level 4: Impression efficiency optimization in recommendation is not carefully considered.** Currently, the issue of how to optimize the impression efficiency for recommender systems has rarely been investigated, though its importance is identified by the evidences of commercial intrusion to users from over-quantity recommendation impression in previous work [21, 33, 71, 99]. Therefore, the issue of how to formulate and optimize the impression efficiency in fairly balancing the utilities between users and E-business companies is in need for investigation.

To solve these limitations, four main pieces of work would be presented in this thesis.

In the first work, we propose a relational recommendation approach for relational fusion of various features, based on conditional random fields (CRF). Different from most previous methods which can only model local features, CRF can model both local and global relational features. In this way, all the predictions of each user-item pair are dependent with each other. Consequently, many user-item pairs which cannot find reliable information before would have such relational features in the prediction. Therefore, the performance would be improved. Multi-scale continuous CRF is utilized in our framework. Traditional single-scale continuous CRF cannot be applied to CF directly, because single-scale CRF can only model the relational features for items of a single user, but fail to model the common behavior patterns of different users. In this work, we extend previous CRF model from single-scale to multi-scale. Then we propose an optimization approach based on Gibbs-sampling. Experimental

verifications demonstrate that the relational features are effective in improving the performance of recommender systems and the framework is effective in multiple features fusion.

In the second work, we investigate methods to fuse regression-oriented and ranking-oriented CF algorithms together to improve performances in both measures. In other domains such as documents/advertisements search tasks, previous work has already indicated that the combination would enhance both performances effectively; however, in collaborative filtering, such combination has never been investigated before. Moreover, previous combination methods for documents/advertisement search cannot be directly employed in CF tasks, because predictions in CF lack effective content features, which is the key point for documents/advertisements search under the classical classification framework. Thus new combination methods should be explored to solve this problem in CF. We propose various combination methods in both model-based and memory-based CF algorithms. Through experimental verification on two real-world datasets, MovieLens and Netflix, we demonstrate that the combinations are effective in improving the performance.

In the third work, we investigate methods to fuse two dimensions, quality and relevance, as a preliminary work for multi-dimensional adaption in recommendation. Such combination has rarely been investigated before. We first give qualitative and quantitative analysis of competitive quality-based and relevance-based algorithms in these two dimensions to show that both algorithms cannot work well in the other dimension. Then we propose an integrated metric and finally investigate how to combine previous work together into an unified model. In the combination, fundamental combination methods suffer from the integration-unnatural and quantity-missing problems. To address these limitations, we introduce a Continuous-time MArkov Process (CMAP) algorithm for ranking, which enables princi-

pled 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 algorithm or relevance-based algorithm in the integrated metric, and the CMAP model outperforms fundamental combination methods.

In the last work, we study the problem of recommendation impression efficiency optimization for less commercial intrusion to users. This issue is important, but is also rarely investigated before. We study this problem under a specific recommendation, sponsored advertisements recommendation in sponsored search. We first formulate the impression efficiency and the problem of impression efficiency optimization under the framework of secretary problem, and then investigate how to solve it. The challenge lies in that the revenue distribution of query-advertisement pairs is not stable and is changing over time. Thus fundamental static methods cannot achieve good performance. In this paper, we propose a dynamic algorithm, which is an extension of previous online algorithm of secretary problem, from 1-tuple multi-choice to 3-tuple multi-choice, in order to solve the unstable problem. Through empirical analysis, the performance of our method outperforms the one of static method significantly, and is approaching the optimal value.

1.2 Thesis Contributions

The main contributions of this thesis are as follows.

1.2.1 Relational Fusion of Multiple Features for Single Measure and Dimension

In this work, we propose a relational fusion approach based on multi-scale continuous CRF and utilize the relational depen-

dency to improve recommender systems. In this approach, relational dependency within predictions is modeled by Markov property. To model various dependency features, we extend previous single-scale continuous CRF to multi-scale continuous CRF. In addition, we propose a gradient-based optimization algorithm to train the model and a constrained simulated annealing process for inference. Gibbs sampling methods in Markov chain Monte Carlo estimation are employed in both training and inference processes to make the inference convergence fast. Through experimental results on two real world datasets, Epinions and MovieLens, we identify that the relational dependency is effective in improving the performances of recommender systems. We also demonstrated that the approach is effective in combining various features.

1.2.2 Effective Fusion of Regression and Ranking for Multi-measure Adaption

In this work, we propose two fusion-based approaches to combine regression and ranking in collaborative filtering. The first one is based on model-based CF algorithms. The combination is based on the joint objective functions from two competitive model-based methods, regression-oriented probabilistic matrix factorization (PMF) [128] and ranking-oriented list-wise matrix factorization (LMF) [139]. The second one is based on memory-based CF algorithms. The combination is based on the joint results from two memory-based methods, regression-oriented user-based Pearson Correlation Coefficient (PCC) method [17] and ranking-oriented EigenRank [91] model. Through experimental verification on two real-world datasets, MovieLens and Netflix, we identify that the combination is effective in improving performances on both metrics.

1.2.3 Effective Fusion of Quality and Relevance for Multi-dimensional Adaption

In this work, we identify that both quality-based algorithms and relevance-based algorithms cannot work well in the other dimension through both qualitative and quantitative analysis. We propose an integrated metric and introduce a large scalable approach CMAP to fuse the dimensions of quality and relevance in multi-dimensional recommendation. The model can solve the integration-unnatural and quantity-missing limitations of fundamental combination methods. Through empirical study on two real world datasets, MovieLens and Netflix, we identify that the combination can significantly outperform single quality-based and relevance-based algorithms in the integrated metric. We also demonstrate that our proposed framework is effective and can outperform fundamental combination methods by around 3%.

1.2.4 Recommendation Impression Efficiency Optimization

In this work, we first formulate the task of optimizing impression efficiency under the secretary framework, which is a classical problem in computing theory. Secondly, through analysis on real world dataset, we identify the observation that the estimated revenue of query-advertisement pairs is not stable, therefore fundamental static method cannot work well. In the third, we propose a novel dynamic approach for the problem. Experimental results show that it outperforms static method significantly. Finally, we combine the static and dynamic approaches, which obtains another improvement.

1.3 Thesis Organization

The organization of the thesis is as follows.

- Chapter 2
  This chapter presents the background and related work of recommender systems. The content would include multiple collaborative filtering methods, multi-measure adaption in recommender systems, multi-dimensional adaption in recommender systems, impression efficiency in recommendation, traditional fusion techniques, relevant applications and competitions.

- Chapter 3
  This chapter presents the proposed relational fusion approach based on multi-scale continuous CRF. We first present the limitation of previous work that many user-item pairs have no reliable information for prediction if relational features are not utilized. Then we introduce how CRF can be utilized in recommender systems. Traditional single-scale continuous CRF cannot be employed directly to model the common behaviors of various users, thus we propose to extend previous CRF from single-scale to multi-scale. After that we describe how to train and inference the model based on Gibbs sampling methods. Experimental verifications are conducted to identify that (1) relational features are effective in improving the performance of recommender systems and (2) CRF is effective in multiple features fusion.

- Chapter 4
  This chapter presents the fusion techniques of regression and ranking for multi-measure adaption in recommender systems. We first present the limitation that either regression-oriented or ranking-oriented algorithms would have over-bias in their own metrics and they also do not fully uti-

lize the observed information. Then we propose how to combine regression and ranking in both model-based and memory-based CF algorithms. Experimental verifications are conducted to show that all the combination methods are effective to improve the performance of recommender system in both metrics.

- Chapter 5
  This chapter presents the fusion model of quality and relevance for multi-dimensional adaption in recommender systems. We first present the problem that single-dimensional recommendation cannot adapt to other dimensions through a detailed data analysis in real world dataset. Then we propose an integrated metric for multi-dimension recommendation as the first ever solution. After that we describe how we combine typical recommendation algorithms in the dimensions of quality and relevance. We propose an framework based on Continuous-time MArkov Process (CMAP) for principled and natural integration with features. Experimental verification are conducted to demonstrate that (1) the combination is effective in multi-measure recommendation and (2) the CMAP consistently outperforms fundamental combination methods.

- Chapter 6
  This chapter presents the work for impression efficiency optimization in sponsored advertisement recommendation. We first formulate the problem under the secretary problem framework. Then we show through statistics in real-world dataset that the estimated revenue of query-advertisement pair is not stable over times, thus static methods would fail to model the unstable problem. After that we propose a dynamic algorithm, extended from previous work. We give empirical study on it. Finally, we combine the static and

dynamic algorithms together to obtain another improvement.

- Chapter 7
  This chapter summarizes the thesis and gives some directions for the future work.

To guarantee each chapter self-contained, some content such as fundamental methods, metrics, etc., may be briefly reviewed in some chapters.

□ **End of chapter.**

Chapter 2

Background Study

2.1 Multiple Collaborative Filtering Methods for Single Measure and Dimension

2.1.1 Classical Regression Problem Definition

Originally, the problem of recommender systems has been formulated as a single problem with regression measure and quality dimension [65]. Over the years, this research issue has been deeply investigated by both industry and academia communities. Let U be the set of all users with User Id u_j, and let I be the set of items with Item Id i_k. A matrix can be built with U and I as the two dimensions, as shown in Fig. 2.1. In this example, there are four users and seven items. Some values of the matrix's elements are fixed from users' rating history R, indicating an quality score on each item (e.g., range from 1 to 5, higher value means better satisfaction.), denoted as r_{jk} in the figure. A large scale of values are missed, denoted as y_{jk}. The problem is to build a function f to predict a score for each missing value of the matrix, formulated as

$$Y = f(R), \tag{2.1}$$

based on which a ranking list of unrated items can be built for each user as recommendation results.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7
u_1	y_{11}	y_{12}	r_{13}	y_{14}	y_{15}	y_{16}	y_{17}
u_2	y_{21}	y_{22}	y_{23}	y_{24}	r_{25}	y_{26}	y_{27}
u_3	y_{31}	y_{32}	y_{33}	y_{34}	y_{35}	y_{36}	r_{37}
u_4	y_{41}	y_{42}	r_{43}	y_{44}	y_{45}	y_{46}	r_{47}

Figure 2.1: User-item matrix in recommendation problem

Metrics for this task measure the closeness of a recommender system's predicted ratings to the users' real ratings. Typical metrics include Mean Absolute Error (MAE) [17] and Root Mean Square Error (RMSE) [76]. MAE is defined as

$$MAE = \frac{\sum |R_{u,i} - \tilde{R}_{u,i}|}{N}, \tag{2.2}$$

where $\tilde{R}_{u,i}$ is the predicted ratings of item i by user u, $R_{u,i}$ is the ground truth, and N is the total number of testing predictions. RMSE is defined as

$$RMSE = \sqrt{\frac{\sum (R_{u,i} - \tilde{R}_{u,i})^2}{N}}. \tag{2.3}$$

In both metrics, lower value indicates higher accuracy.

2.1.2 Collaborative Filtering Techniques Overview

We summarize classical recommendation techniques in Fig. 2.2.

Before collaborative filtering, content-based methods are utilized in recommender systems. In content-based methods [82], items with similar content features comparing to a user's past favorite items will be recommended to the user. For example [104],

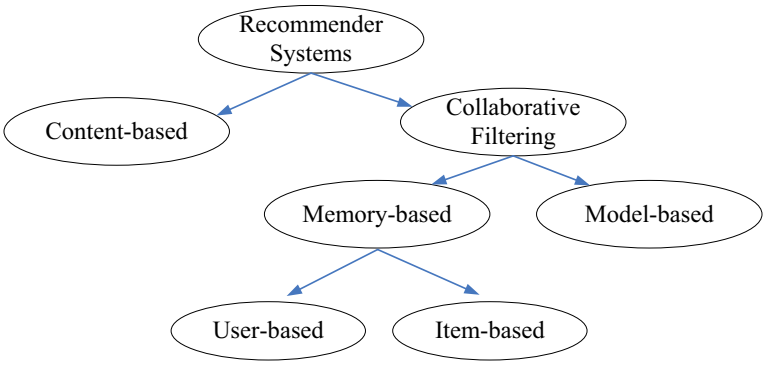


Figure 2.2: An overview of recommendation techniques

in movie recommendation systems, such features include specific actors, directors, genres, subjects, etc. Typical work includes [10, 106, 107, 109, 117, 126, 144, 168, 169].

The weakness of this kind of methods is that it depends on the features, and effective features are difficult to find in some recommendation applications. For example, in the Amazon, many users have very incomplete profiling information, and the items in their history have a quantity of diversity. Thus there are not enough features for accurate predictions. This is the reason that collaborative filtering comes up.

Collaborative filtering (CF) methods are from a different angle. The prediction is based on the common behavior patterns analyzed from the large real dataset. The key point is that CF finds similar users for each user, according to the similarity of their rating history. Then the prediction is made by the ratings of his/her similar users. Over the years, this kind of technique has made a great success and has been deeply investigated. Traditionally, there are two classes of CF approaches: (1) memory-based and (2) model-based.

The idea of memory-based methods (also called neighborhood-based) is that the rating predictions for a user directly depend on his/her similar users' ratings on similar items. Approaches are further divided into two streams, user-based [17, 64] and item-

based [88, 130]. In user-based algorithms, the prediction of an item is based on his/her similar users' ratings on it. Item-based algorithms are very similar with user-based algorithms. But it will find similar items for each item according to the similarities of its user. Thus the prediction of an item for a user is based on the user's ratings on its similar items.

In model-based methods [67, 152], the common patterns of users and items are modeled indirectly by latent features. This kind of method builds a generated model from a probabilistic perspective. Latent features are learned through a training process, and the predictions are made by these learned features. According to different objective functions, model-based CF algorithms are further divided into log-likelihood-targeted methods [67] and root-mean-square-targeted methods [128]. Typical work includes [2, 16, 17, 25, 37, 51, 54, 84, 140, 97, 101, 115, 122, 137, 140, 152, 158, 170].

In the following sections, we will give some examples for memory-based CF algorithms and model-based CF algorithms.

2.1.3 Memory-based Collaborative Filtering

The key point of memory-based CF is the similarity selection among users and items. Typical examples include Pearson Correlation Coefficient (PCC) [123] and vector similarity (VS) [17]. In the following part, we explain a user-based algorithm and an item-based algorithm [17] based on PCC similarity.

User-based PCC

User-based PCC is defined as [96]

$$Sim_{a,u} = \frac{\sum_{i \in I(a) \cap I(u)} (r_{a,i} - \overline{r}_a)(r_{u,i} - \overline{r}_u)}{\sqrt{\sum_{i \in I(a) \cap I(u)} (r_{a,i} - \overline{r}_a)^2} \sqrt{\sum_{i \in I(a) \cap I(u)} (r_{u,i} - \overline{r}_u)^2}} \quad (2.4)$$

where a and u denote two users, $I(a)$ and $I(u)$ are the items they have rated, $r_{a,i}$ is the rating of item i by user a, and $\overline{r}_a$ is the average rating of user a.

The advantage of this similarity is that it balances different users' evaluation standards. For example, some users are more likely to give high ratings and other may like to give low ratings. The similarity is based on the difference of values rather than absolute values.

By using PCC similarity, we can find users with high similarity to the current user as its neighborhood. Then, the rating predictions are based on the following formula,

$$f(u,i) = \overline{u} + \frac{\sum_{u_a \in S(u)} Sim_{u_a,u}(r_{u_a,i} - \overline{u}_a)}{\sum_{u_a \in S(u)} Sim_{u_a,u}}, \tag{2.5}$$

where $S(u)$ is the neighborhood of the current user u, $\overline{u}$ is the user's average rating score.

Item-based PCC

The idea of item-based PCC is very similar to user-based PCC. The difference is that user-based PCC finds neighbors for each user, but item-based PCC finds neighbors for each item. Thus item-based PCC is defined as [96]

$$Sim_{i,j} = \frac{\sum_{u \in U(i) \cap U(j)} (r_{u,i} - \overline{r}_i)(r_{u,j} - \overline{r}_j)}{\sqrt{\sum_{u \in U(i) \cap U(j)} (r_{u,i} - \overline{r}_i)^2} \sqrt{\sum_{u \in U(i) \cap U(j)} (r_{u,j} - \overline{r}_j)^2}} \tag{2.6}$$

where i and j are two items, $U(i) \cap U(j)$ denote the users who has rated both i and j, and $\overline{r}_i$ is the average rating of item i.

Under item-based PCC, the prediction is based on

$$f(u,i) = \overline{i} + \frac{\sum_{i_k \in S(i)} Sim_{i_k,i}(r_{u,i_k} - \overline{i}_k)}{\sum_{i_k \in S(i)} Sim_{i_k,i}}, \tag{2.7}$$

where $S(i)$ is the neighborhood of current item i, $\overline{i}$ is the item's average rating score.

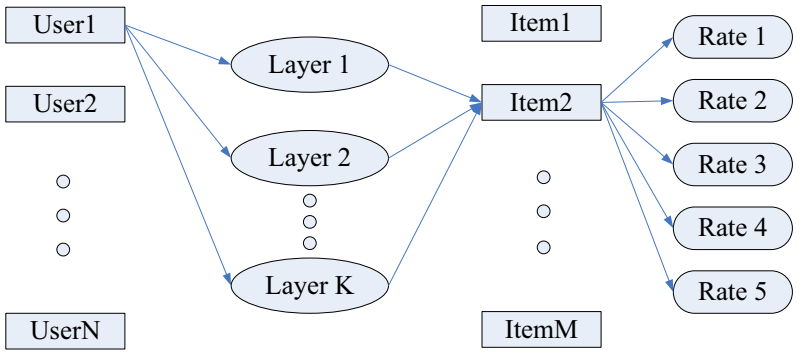


Figure 2.3: Single-direction aspect model

2.1.4 Model-based Collaborative Filtering

Model-based collaborative filtering is another kind of typical collaborative filtering method. The main difference from memory-based methods is that memory-based methods should load all the ratings in the memory when predicting, while model-based methods try to learn a model from existing data, and use the model in prediction without loading the rating information. In the following, we will use two examples to show model-based algorithms, (1) Aspect Model (AM) [67] and (2) Probabilistic Matrix Factorization (PMF) [128]. Comparing to memory-based methods, model-based collaborative filtering methods save more memory and complexity in predictions. However, to train a model needs much time.

Aspect Model

As shown in Fig. 2.3, in Aspect Model [67], latent layers (classes) exist between users and items, which can be explained as the users' interests or styles. Different latent layers have different distributions on the rating of items, and different users have different distributions on the latent layers. These distributions are learned from training data by optimizing the log-likelihood. Expectations are calculated as predictions. For example, the

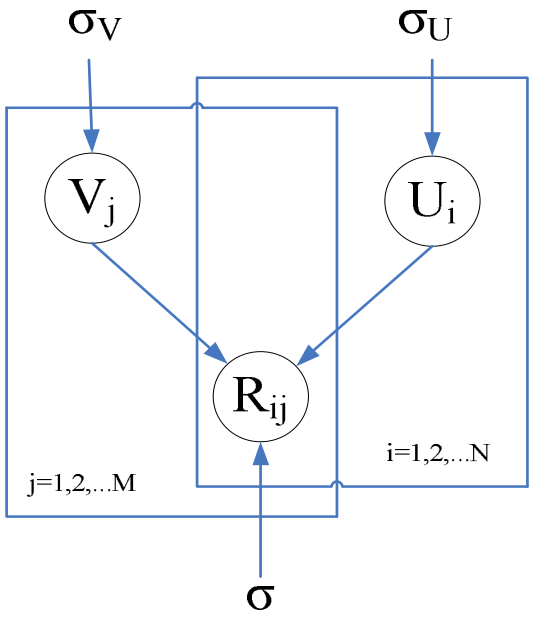


Figure 2.4: Probabilistic graph of probabilistic matrix factorization

prediction of item k by user j is calculated by

$$r_{j,k} = \sum_{l=1}^{K}(\sum_{r=1}^{5} r * p(r|l))P(l|j). \tag{2.8}$$

Probabilistic Matrix Factorization

Fig. 2.4 shows the probabilistic graph of PMF. Suppose there are N users and M items. For each user and item, there is an l-dimensional latent feature vector. The feature vectors for users are denoted as $U \in R^{l*N}$ and the feature vectors for items are denoted as $V \in R^{l*M}$. Let R_{ij} denote the rating of item j given by user i. In this graph, the distribution of R_{ij} is defined as [128]

$$P(R|U, V, \sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{M}[N(R_{ij}|g(U_i^T V_j), \sigma^2)]^{I_{ij}}, \tag{2.9}$$

where $N(x|u, \sigma^2)$ is a Gaussian distribution with the mean u and variance σ^2. $g(x)$ is the logistic function $g(x) = 1/(1+\exp(-x))$

to convert $U_i^T V_j$ to $[0, 1]$ scale. I_{ij} is an indicator to describe whether user i has rated item j.

The objective function of PMF is to find U, V by minimizing the summation of regression loss and regularization as

$$\arg\min_{U,V} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} I_{ij}(R_{ij} - g(U_i^T V_j))^2 + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2). \quad (2.10)$$

2.1.5 More Machine Learning Techniques for Collaborative Filtering

Besides the fundamental collaborative filtering approaches, there are also a number of relevant machine learning techniques as complement for recommender systems.

Active Learning for Collaborative Filtering

Usually, recommender systems suffer from the "cold start" problem. It means that for a new item or a new user, there is no rating history for it. Thus it is very difficult for collaborative filtering to make prediction on these new items or users.

A typical way to solve this problem is to let the new user rate some items. The importance of each item is different. Thus the target of active learning for collaborative filtering is to find the most important items to the new user to rate, in order to get the most effective collaborative filtering model for the user. Some typical work includes [61].

Online Learning for Collaborative Filtering

Online learning is to adapt to instant update information for recommender systems. The original mode of training a collaborative filtering model is a batch mode. It means there is a training set to learn the model, and in the application, the model

would be fixed. However, in the application, some new information may come into the system (e.g., a user may rate a new item). The problem is that these new information would not be utilized to improve the model until the next batch learning process. Thus the accuracy would be limited.

Online learning is to develop algorithms that can update the model instantly as the new information come into the system. Online learning algorithms will remain most of original model stable while make a little change for the new information. Thus the complexity for updating the model is much less than training a new model. The updating will have an improvement for current model to fit the new data. Typical work includes [3].

Others

There are also other machine learning techniques for recommender systems. For example, some work tries to speed up the learning process of collaborative filtering models [3, 34, 73, 166]; some work tries to make the model scalable to large dataset [34, 89, 136, 165, 174]; some work tries to apply transfer learning technique in collaborative filtering [85]; and many others can be found in [27, 103, 129, 146, 161].

2.2 Multi-measure Adaption for Recommender Systems

The classical task of recommender systems is the regression-oriented problem. However, as the final output of recommender systems is a ranking list of items for users, the ranking-oriented recommendation has been paid much attention recently. In the ranking-oriented recommendation, measurement is from a ranking perspective. The two measurements are quite different for adaption. Ideally, a good recommendation algorithm should

have good performances adapting to both kinds of measurements.

2.2.1 Ranking-adapted Recommendation

Ranking-oriented recommendation is proposed recently [91]. The goal is to predict the ranking order of items for each user directly, where the rating predictions are not necessary [91]. Thus the ranking relation information is directly modeled and the performance is evaluated on the ranking order of predicted items. Ranking-oriented metrics measure the closeness of predicted ranking with the ground truth ranking. Typical metric is the Normalized Discount Cumulated Gain (NDCG) [32] value. Given the rank of recommended results, NDCG at position P is defined as (referring to [91])

$$NDCG_{P-quality} = \frac{1}{U}\sum_{u}^{U} Z_u \sum_{p=1}^{P} \frac{2^{r_{u,p}} - 1}{\log(1+p)}, \tag{2.11}$$

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. For NDCG, a larger value indicates the higher accuracy.

Collaborative filtering approaches can be employed to solve ranking-oriented recommendation by changing relevant key calculation (e.g., similarity, loss function, etc.) to ranking-oriented. Currently, the algorithms are further divided into model-based [92, 139] and memory-based algorithms [91]. Other work includes [38, 94]

2.2.2 Ranking v.s. Regression

Both regression-oriented and ranking-adapted recommendations have their own advantages. The advantage of regression-oriented

algorithms is that the prediction is more intuitive. The ratings are easier than rankings to understand for the evaluation on items. For example, we can easily calculate the average ratings for the items. We can also understand the difference between two items by their difference on the ratings. In addition, the complexity to evaluate a rating is usually much less than to evaluate a ranking. The advantage of ranking-oriented algorithms, on the other hand, is that the information is richer, especially when the data is sparse. Suppose a user have four ratings. In regression-oriented recommendation, it is not confident to utilize the comparison of the ratings to find similar users. However, if the ranking information is utilized, four ratings have six preference orders, which is more reliable. From the features of regression and ranking, it can be concluded that both are important to the success of recommender systems.

2.3 Multi-dimensional Adaption for Recommender Systems

Traditionally, the performance of recommender systems is from the quality dimension. The quality is described by a rating. The task is to recommend high-quality items to users. However, quality is only one dimension in the performance of recommender systems. There are many other dimensions that users also concern about. Such dimensions include relevance, diversity, coverage, etc. A good recommendation should consider all these dimensions according to the user's configurations.

2.3.1 Relevance Dimension Adaption

Relevance is a different dimension from the dimension 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.

The dimension of relevance in recommender systems has been paid much attention recently. Relevance-based recommendations mainly depend on association features [36, 131] and hitting frequency features [12, 80, 149]. The basic assumption of the former is that frequent co-occurred 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 [80]. Recent work of relevance-based recommendations includes [113, 158].

Relevance-based metrics measure the likelihood that an item will be hit. Metrics are also divided into regression-oriented and ranking-oriented. The difference from quality is that the relevance score is a 0/1 value instead of the rating from 1 to 5.

2.3.2 Other Dimensions Adaption

Besides the dimensions of quality and relevance, there are many other dimensions for the performance of recommender systems.

Diversity [46] and coverage [154] are two typical dimensions. Coverage means to what extend the recommendation can cover all the items; and diversity means how different the recommended items are from each other. Typical work on these dimensions includes [72, 83, 111, 175].

2.4 Recommendation Impression Efficiency

Besides the performance, there are many more issues for consideration in the success of recommender systems. A very fundamental issue is the recommendation impression efficiency. Impression efficiency describes how much profit can be obtained per impression of recommendation result.

The reason for optimizing impression efficiency is that over-quantity of recommendation result would have commercial intrusion to users. Since most recommender systems are supported by E-business companies, the recommendation can be seen as a commercial behavior from E-business companies to users. As the E-business and recommender systems become popular, there are many evidences indicating the existence of commercial intrusion to users from over-quantity recommendation, especially in sponsored advertisement recommendation in sponsored search: 1) Users have reported to show bias against sponsored search results after they know its commercial insight [99]. 2) From the user study in [71], when sponsored results are as relevant as the organic results, more than 82% of users will see organic results first. 3) Organic results have also demonstrated to gain much higher click through rate (CTR) than sponsored search results [33].

Thus if the impression efficiency is not carefully optimized, in a long-term, users will not trust recommender systems due to the commercial intrusion and finally it will decrease the utility of E-business. If the recommendations are irrelevant to users'

intent, to show less recommendation results or even not to show any recommendation results is better than to show a full rank of results [18]. From the research in [21], irrelevant recommendation will have the effect to "train" the users to ignore recommendation in the result page.

Although the intrusion of recommender systems to users has already been identified from previous work, there is rarely much work carefully investigating how to optimize the recommendation impression efficiency in previous work. Thus in later chapters, we will investigate this problem as a preliminary work.

2.5 Traditional Fusion Techniques Overview for Recommender Systems

With the development of different algorithms of collaborate filtering, there are also many fusion work to combine many components together to get better performance. The fusion work can be divided into fusion of various kinds of information and fusion of various algorithms.

2.5.1 Fusion of Various Information

This kind of fusion work utilizes additional information beyond the user-item rating matrix to enhance the performance of recommender systems. Typical information includes social relationship of users, temporal information, location information, etc. We will explain some of them as follows. Other work of this kind of fusion includes [4, 14, 59, 98, 102, 104, 108, 134, 142, 153, 156].

Social Information Ensemble

Users on the Web are not alone. Usually, many relationships exist among users. Such relationships can be trust link, friend

link, twitter link, etc. The assumption of utilizing social information to enhance recommender systems is that users connected by these links would have similar tastes on items. For example, if two users are good friends, they may like the same products. From previous work, effectively incorporating the social information into collaborative filtering models would obtain significant improvements. Typical work includes [5, 28, 110, 53, 69, 75, 97].

Temporal Information Ensemble

Temporal information is another typical feature in recommender systems. The assumption is that users' tastes may change over the time. For example, a user may be very critical at the beginning, but after some time, he/she may be changed to give higher ratings to most items. Thus the temporal information is considered in collaborative filtering methods. Through experimental verification, such information is also very effective in improving the performance. Typical work includes [70, 77, 159].

Location Information Ensemble

As applications on mobile are more and more popular, location information becomes important in recommender systems. For example, if a user is searching for a restaurant, he/she may prefer the ones near him/her. Thus recently, some collaborative filtering approaches are proposed to incorporate the location information in recommendation. Typical work includes [50, 171]

2.5.2 Fusion of Various Algorithms

Over the years, many algorithms have been proposed for collaborative filtering [68]. Each algorithm has its own advantages and disadvantages. Thus it is natural to combine them together to get the best performance. The fusion work for various algorithms can be divided into: (1) combination of user-based and

item-based in memory-based algorithms [96, 157, 163]; (2) combination of memory-based and model-based algorithms [76, 118]; and (3) combination of content-based and collaborative filtering [10, 12, 30, 104, 116, 132, 143, 147].

There are mainly two limitations from previous work. The first one is that the relational dependency among features and predictions is not utilized in the fusion. Thus the performance would be limited when the data is sparse. The second one is that all the fusion work is to solve the classical regression problem for recommender systems. However, as discussed in the introduction chapter, there are mainly four levels to evaluate recommender systems, such as multi-measure adaption, multi-dimensional, impression efficiency, etc. Fusion work on these levels has rarely been investigated before. This is the target of the thesis.

2.6 Applications and Competitions Related to Recommender Systems

2.6.1 Applications

A number of recommender system applications appear with the development of recommendation techniques. These applications cover many aspects of our daily lives, such as music, web page, books, etc. They also experience different methods. We summarize some typical recommendation systems in Table 2.1.

Besides these systems, there are also many algorithms developed to specific recommendation domains, such as tags [56, 121, 141, 135, 145], communities [24, 23], citations [62], news [86, 95], documents [57, 173], queries [150], etc.

Table 2.1: Recommender system applications

System	Content	Techniques
Amazon [88]	books, CDs, others	item-based
MovieLens [105]	movie	item-based
Grundy [124]	books	content-based
Video Recommendar [66]	video	memory-based
Ringo [138]	music	user-based
PHOAKS [151]	textual information	memory-based
Jester [54]	jokes	model-based
Fab System [10]	Web page	hybrid approaches

2.6.2 Competitions

In this section, we will briefly introduce three big competition events for recommender systems.

Netflix

Netflix competition[1] started from October 2006, which was conducted by the DVD renting company Netflix[2]. The goal is to design the best recommendation algorithms to recommend movies to users. The task is to predict the ratings of users on items as close as possible (the ratings are from 1 to 5 with higher value indicating better satisfaction). The evaluation metric is RMSE (the lower the value, the better the performance). Originally, the RMSE of the algorithm in Netflix was 0.9525; and anyone that could improve it by 10% (0.8572) would win the 1,000,000 Grand Prize.

Comparing to traditional datasets, there are two main challenges: (1) the dataset is large-scale. Comparing to a classical MovieLens datasets[3] with 100,000 ratings within 1,682 items and 943 ratings, Netflix dataset has 100,000,000 ratings from

[1]http://www.netflixprize.com/
[2]http://www.netflix.com/
[3]http://www.movielens.org/

over 480,000 users for 17,000 movies. This means many complex graphical models cannot be directly utilized in such a big application. (2) The data is very sparse. The density of the user-item matrix in Netflix is 1.18%, compared with 6.3% in MovieLens. This means that most ratings have not enough reliable information for predictions. Thus the competition was challenging.

Over five years, the team "BellKor's Pragmatic Chaos" finally won the prize. Their work is an effective combination of previous collaborative filtering methods. Detailed algorithms can be found in their publication [76].

KDD Cup 2007

The KDD Cup 2007[4] is a competition related to Netflix. They share the same dataset, but the tasks are different. The original criterion of Netflix is focusing on the quality of items; but the criteria in this KDD Cup are related to the relevance of items.

There are two subtasks with different metrics. The first task is called "Who Rate What in 2006", which is to predict which movies a user will be likely to rate in 2006 according to the history information from 1998 to 2005. The second task is called "How Many Ratings in 2006", which is to predict how many movies a user will rate in 2006 according to the history information from 1998 to 2005. The metrics for both tasks are RMSE.

At the end, the team of Hungarian Academy of Sciences won the first place in the first task and the team of IBM Research won the first place in the second task. The algorithms are effective fusions of previous memory-based and model-based methods. More details can be found in their reports [80, 127].

[4]http://www.cs.uic.edu/ liub/Netflix-KDD-Cup-2007.html

KDD Cup 2011

The KDD Cup 2011[5] is a recent competition on recommender systems. The dataset[6] is from Yahoo! music recommendation. Different from previous movie recommendation, this dataset contains structural information. Users can rate tracks, albums, artists and genres. The tracks can be structured by albums, artists, and genres. Thus the challenge is how to utilize such structural information in improving the recommendation performance. In this dataset, it has 1,000,990 users, 624,961 items and 262,810,175 ratings. Comparing with previous dataset, the number of items has increased greatly.

There are totally two tasks. The first task is to predict the users' ratings to the items. The items contain all four kinds of information (tracks, albums, artists, and genres). RMSE is utilized as the evaluation metric. It is a classical regression problem. The second task is to identify the highly-rated items from the others for a user. The difference is that it is a classical classification problem. The evaluation metric is the error rate (fraction of misclassifications).

Currently, the competition is still open and will end in the end of June 2011.

□ **End of chapter.**

[5] http://kddcup.yahoo.com/
[6] http://new.music.yahoo.com/

Chapter 3

Relational Fusion of Multiple Features for Single Measure and Dimension

3.1 Limitations of Previous Fusion Techniques

In recommender systems, multiple features should be utilized to improve recommendation results. Traditional collaborative filtering (CF) algorithms, however, suffer from the following two weaknesses.

To illustrate the problem, we use an example showed in Fig. 3.1. In this example, there are four users, denoted by u_l and seven items, denoted by i_m. r_{lm} is rating record by u_l to i_m. (e.g., scale from 1 to 5, higher value means better satisfaction). The CF algorithms predict values of unrated user-item pairs, denoted as y_{lm} (without loss of generality, not all y_{lm} are shown in the figure), and suggest top ranked items as recommendations.

Lack of relational dependency within predictions. In traditional methods, predictions are only relationally dependent on the rated records, while predictions among each other are independent. For example, in Fig. 3.1, suppose u_3 and u_4 are similar users based on the observed ratings, and then y_{33} can be predicted by referring to r_{43}, because it is the same item and

the two users have high similarity. In the same way, suppose i_3 and i_5 are observed to have high similarity, and then y_{45} can be predicted by referring to r_{43}, because they are similar items by the same user. For simplicity, we suppose no high similarity exists between other items/users pairs, and we do not consider any other relations. In this case, based on traditional CF algorithms, y_{35} cannot be predicted accurately, because there are no rated items by u_3 which is similar to i_5 and there is no rating on i_5 whose host is similar to u_3. Thus no relevant information can be referred to. But if we consider relational dependency within predictions, things are different. As u_3 and u_4 are similar, y_{35} and y_{45} should be close; as i_3 and i_5 are similar, y_{35} and y_{33} should be close. So if relational dependency within predictions is utilized, the information of r_{43} can be passed to y_{35} through relational dependency of y_{33}, y_{45}, and y_{35}. In this case, predictions should be generated simultaneously by utilizing the dependency, which let predictions help each other, improving the accuracy. In recommender systems, the data is sparse [130], thus a number of predictions lack of information to refer to, leading to low accuracy. Effectively utilizing relational dependency is indeed important. Previous work, however, did not utilize such information sufficiently. Wang et al. [157] proposed a heuristic method to find r_{43}. It has two limitations: (1) It is difficult to measure the similarity between r_{43} and y_{35}; and (2) It cannot guarantee the closeness of y_{35} and y_{33} (or y_{35} and y_{45}). Ma et al. [96] proposed to firstly predict y_{33} and y_{45}, and then to predict y_{35}. The problem is that mistakes can propagate from the top level to the bottom level, which influences the accuracy.

Being difficult to integrate various features in social network into an unified approach. In recommendation, various attribute information and relations have been demonstrated to be effective features. For example, in attribute information, Melville et al. [104] utilized content information (gen-

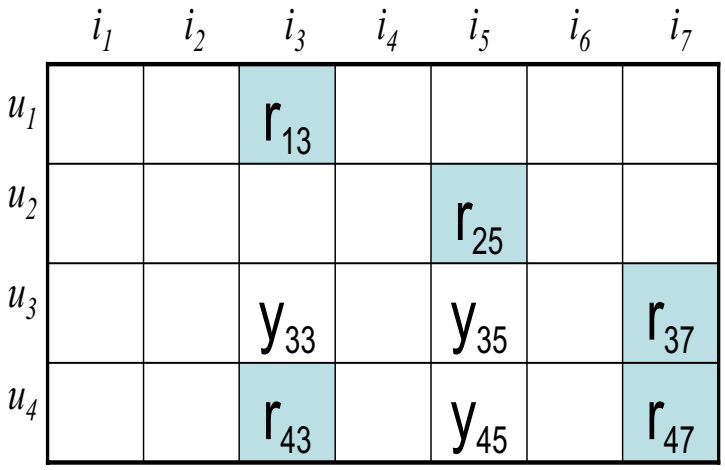


Figure 3.1: An illustration example to show the limitations of traditional methods

res, directors, etc.) to boost CF algorithms in movie recommender systems; Nakamoto et al. [108] and Sen et al. [134] employed tag information to improve the accuracy. In relations information, trust relations are utilized effectively in some recent works [5, 13, 53, 97]. These attribute and relation features should be combined to assist predictions in relational recommendation. But in traditional CF algorithms, it is hard to combine these features into an unified model. Melville et al. [104] has to convert traditional CF to a classification problem in order to add content features, in which ratings are not predicted. Some of previous work utilized linear integration techniques to smooth feature weights [96]. Consequently, the computing complexity for enumerating values in all spaces to obtain a fitting weight-vector is large when the number of features increases. Thus a framework to globally optimize (optimize all the weights simultaneously) the weights of multiple effective features should be explored.

3.2 Conditional Random Fields for Recommender Systems

Conditional Random Fields (CRF) is first proposed as a state-of-the-art probabilistic model for segment and labeling sequences data [60, 81]. This model can describe relational dependency in undirected probabilistic graphs, solving the label bias problem. Due to effectiveness in many applications, the theory is widely developed such as Multi-scale CRF [63], Constrained CRF [78, 160], etc. A more detailed tutorial can be found in [44]. Qin et al. [119] first extended conditional random fields from discrete label spaces to continuous label spaces, and applied this CCRF model to rank documents.

Continuous Conditional Random Fields (CCRF) [119] is a desirable approach by going through literatures on solving similar problems mentioned above. Relational dependency within predictions is modeled in feature functions in CCRF. CCRF has outstanding advantages comparing to other methods: (1) relational dependency within predictions can be modeled by the Markov property, which is the most general assumption in probabilistic graphical models and has been proven effective in many applications [81]; and (2) feature function weights are globally optimized in CCRF model, which makes it easy to combine various of features. Thus all the two problems aforementioned can be solved based on this approach. Therefore, it is natural to lead us to employ CCRF in relational recommendation problems. However, single-scale of CCRF in [119] cannot be directly employed to model different users in recommendations, which will be discussed in detail later. Therefore in this work, we extend CCRF model from single-scale to multi-scale in theory, in which each scale corresponds to predictions of a particular user, and apply this new model in relational recommendations as a framework to solve the two problems discussed above, which to

the best of our knowledge is the first attempt to employ CCRF in recommender systems.

3.3 Relational Fusion Approach for Recommender Systems

3.3.1 Relational Recommendation Formulation

Let X denote observations, which can be existing rating records, trust information, similarities between different users/items, profile information of users, etc. Let vector Y denote predictions with y_{lm} denoting the prediction of item i_m by user u_l.

We call "local recommendation" or "traditional recommendation", if the problem is formulated as

$$y_{l,m} = f(X). \tag{3.1}$$

Further more, we call "global recommendation" or "relational recommendation", if the problem is formulated as

$$\begin{aligned} Y &= f(X), \ or \\ y_{l,m} &= f(X, y_{-l,-m}), \end{aligned} \tag{3.2}$$

where $y_{-l,-m}$ denotes all other predictions except $y_{l,m}$.

The major difference of these two formulations is that predictions in relational recommendation are dependent on each other conditioned on observations and thus predictions on different items should be generated simultaneously; while in traditional recommendation, predictions are independent. In other words, traditional recommendation is a special case of relational recommendation when relational dependency within predictions is removed.

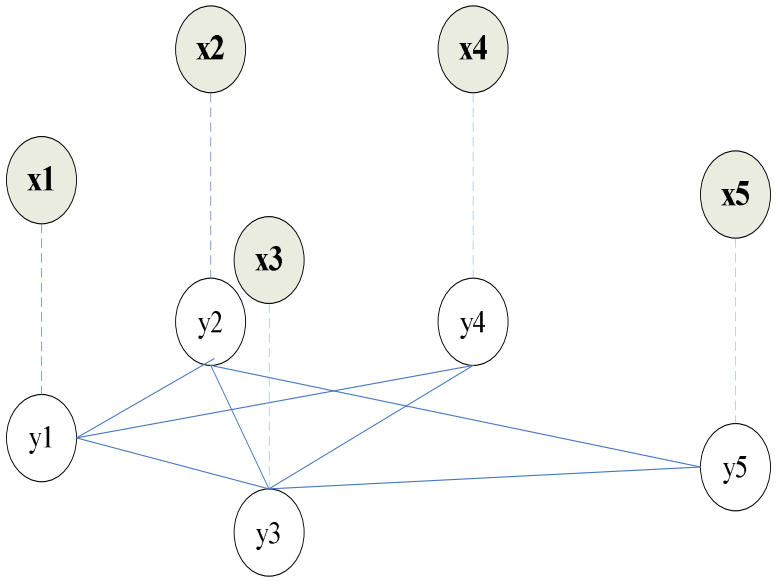


Figure 3.2: Probabilistic graph of single-scale continuous conditional random fields

3.3.2 Single-scale Continuous Conditional Random Fields Fusion Approach

Single-scale CCRF is proposed by Qin et al. [119], applied in the issue of "global ranking". In this model, a joint conditional probability distribution of a probabilistic graph is defined conditioned on observations. In this section, we explain the model in the application of recommender systems. Please notice single-scale CCRF can only model predictions of a single user and we discuss how to handle multiple users in the next sub-section.

The detailed definition of single-scale CCRF is as follows. Figure 3.2 gives the probabilistic graph. Let nodes $X(x_1, x_2, ..., x_5)$ denote observations and nodes $Y(y_1, y_2, ..., y_5)$ denote predictions (y_m for item i_m). The edge connecting y_m and y_n indicates that relational dependency exits between them in the model. We define the set of nodes connected to y_m by actual line as the "neighbor" of y_m, denoted as $neighbor(y_m)$. Since X denotes observations and all values of Y are conditioned on it, we use

dotted line to approximately express the relational dependency among X and Y. The joint conditional probability density function of predictions Y conditioned on observations X is defined as

$$p(Y|X) = \frac{1}{Z_{sgl}(X)} \exp\left\{ \sum_m \alpha \cdot H(y_m, X) + \sum_{m,n} \beta \cdot G(y_m, y_n, X) \right\}, \quad (3.3)$$

where $H(y_m, X)$ is a local state feature functions vector defined on a local value y_m, and $G(y_m, y_n, X)$ is a relational edge feature functions vector defined on the relational dependent values of y_m and y_n. α and β are function weights vectors to be learned from the training dataset. $Z_{sgl}(X)$ is a normalization factor defined as

$$Z_{sgl}(X) = \int_y \exp\left\{ \sum_m \alpha \cdot H(y_m, X) + \sum_{m,n} \beta \cdot G(y_m, y_n, X) \right\} dy. \quad (3.4)$$

The goal for relational recommendation is to find a vector of predictions Y for this user, which can maximize the joint conditional probabilistic distribution of $p(Y|X)$. The feature functions are defined in the quadratic form as:

$$h_{t1}(y_m, X) = -(y_m - x_{m,t1})^2, \quad (3.5)$$

$$g_{t2}(y_m, y_n, X) = -\frac{1}{2} M_{m,n,t2} (y_m - y_n)^2. \quad (3.6)$$

In the equations, t_1 is state feature function index ranging from 1 to T_1 and t_2 is edge feature function index ranging from 1 to T_2. Here, $x_{m,t1}$ is observed features on item i_m, which can be the average rating of i_m; $M_{m,n,t2}$ is a relational feature measure, which

can be the similarity between item i_m and item i_n. If we use these two features as an example, it is not difficult to conclude that $p(Y|X)$ will be high if predictions Y fit the following conditions: (1) predictions on item i_m is close to the average rating of item i_m; and (2) similar items receive similar ratings predictions. Therefore, relational dependency within predictions for a particular user is described in single-scale CCRF model.

3.3.3 Multi-scale Continuous Conditional Random Fields Fusion Approach

Single-scale CCRF cannot model multiple users, because there is only single value for each item, though conditioned relational dependency within predictions is modeled on different items. In this case, all users will be treated the same, which is not reasonable. Besides, what we need to do is not only distinguishing prediction strategies of different users, but also modeling the relational dependency within them. In relational recommendation, various relationships (trust information, similarity information, etc) among users are needed to be modeled. Therefore, in this work, we extend CCRF from single-scale to multi-scale to form a novel model and apply it as a framework in relational recommendations to solve aforementioned limitation.

Figure 3.3 gives the probabilistic graph of MCCRF. In this graph, label space of Y has been extended from single-scale to multi-scale with $y_{l,m}$ denoting prediction on item i_m by user u_l. Different scales of Y are drawn in different layers which denote predictions of multiple users. For example, $(y_{11}, y_{12}, y_{13}, y_{14}, y_{15})$ is the rating predictions for user u_1, and $(y_{21}, y_{22}, y_{23}, y_{24}, y_{25})$ is for user u_2. We still use actual line to denote the relational dependency of predictions Y in the model. In MCCRF, relational dependency exists not only within predictions of the same user (layer), but also within predictions among different

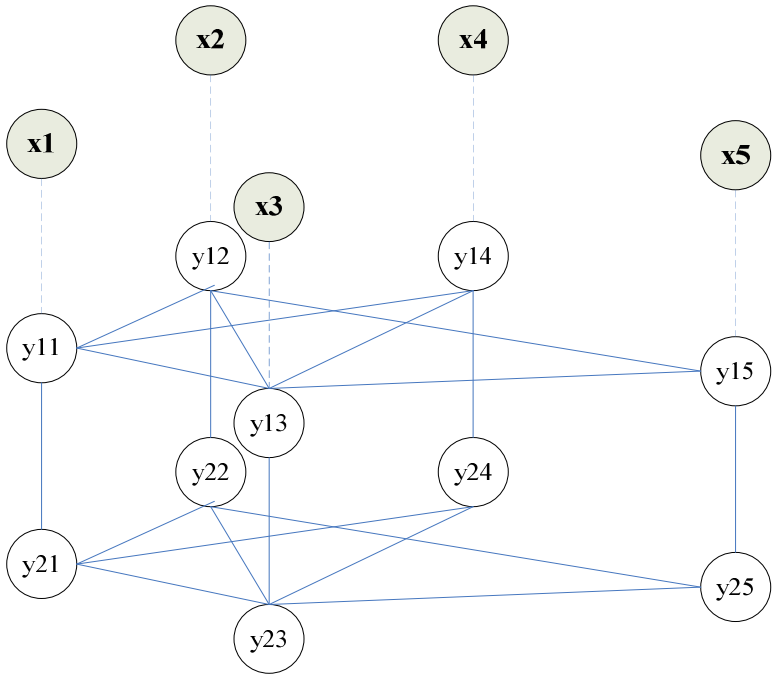


Figure 3.3: Probabilistic graph of multi-scale continuous conditional random fields

users (layers). For example, the prediction of y_{13}, has dependent relationship with $\{y_{11}, y_{12}, y_{14}, y_{15}, y_{23}\}$. This example also shows how $neighbor(y_{13})$ (the five dependent nodes) is defined in MCCRF.

In this model, the joint conditional probability density function is defined as

$$\begin{aligned} p(Y|X) &= \frac{1}{Z_{mul}(X)} \exp\left\{ \sum_{l} \sum_{m} \alpha \cdot H(y_{l,m}, X) \right. \\ &\quad + \sum_{l} \sum_{m,n} \beta \cdot G(y_{l,m}, y_{l,n}, X) \\ &\quad \left. + \sum_{m} \sum_{l,j} \gamma \cdot R(y_{l,m}, y_{j,m}, X) \right\}, \end{aligned} \tag{3.7}$$

where l and j denote different users; m and n denote different items. $H(y_{l,m}, X)$ is a local state feature functions vector

defined on local value $y_{l,m}$; $G(y_{l,m}, y_{j,n}, X)$ is a relational edge feature functions vector defined on relational dependent values within the same layer; $R(y_{l,m}, y_{j,m}, X)$ is a relational edge feature functions vector defined on relational dependent values across different layers. $\{\alpha, \beta, \gamma\}$ is feature function weights vectors to be learned from training data. $Z_{mul}(X)$ is the normalization factor defined as

$$\begin{aligned} Z_{mul}(X) &= \int_y \exp\left\{ \sum_l \sum_m \alpha \cdot H(y_{l,m}, X) \right. \\ &\quad + \sum_l \sum_{m,n} \beta \cdot G(y_{l,m}, y_{l,n}, X) \\ &\quad \left. + \sum_m \sum_{l,j} \gamma \cdot R(y_{l,m}, y_{j,m}, X) \right\} dy. \end{aligned} \tag{3.8}$$

The task for relational recommendations under this framework is to find the predictions Y that can maximize the joint probabilistic distributions $p(Y|X)$. Feature functions are still defined in the quadratic form as:

$$h_{t1}(y_{l,m}, X) = -(y_{l,m} - x_{l,m,t1})^2, \tag{3.9}$$

$$g_{t2}(y_{l,m}, y_{l,n}, X) = -\frac{1}{2} M_{m,n,t2} (y_{l,m} - y_{l,n})^2, \tag{3.10}$$

$$r_{t_3}(y_{l,m}, y_{j,m}, X) = -\frac{1}{2} U_{l,j,t3} (y_{l,m} - y_{j,m})^2. \tag{3.11}$$

Here, $x_{l,m,t1}$ is observed features of i_m or u_l, which can be the average rating of u_l; $M_{m,n,t2}$ is a measure of relational feature in the same layer which can be the similarity of i_m and i_n; $U_{l,j,t3}$ is a measure of relational feature across different layers which can be the trust relation of u_l and u_j (e.g. the value of $U_{l,j,t3}$ is 1 of u_l trust u_j and is 0 of not). Under this definition of features as an example, it is not difficult to conclude that $p(Y|X)$ will be high if Y fits the following conditions: (1) predictions of a

user are close to average rating of the user; (2) predictions on similar items for the same user are close; and (3) predictions of trusted users on the same item are close. Therefore all kinds of relational dependency within predictions have been modeled.

3.3.4 Features for Fusion

The feature selection in our work is experiment-based. In CRF, features are divided into state features and edge features. Following are the features combined in our model. We will also show the effectiveness of each feature in experimental section.

State Features (The three kinds of state features are only provided in MovieLens dataset):

1. Average rating of an item within users of similar occupation.
2. Average rating of an item within users of similar age and same gender.
3. Average rating of the same genre.

Edge Features (Trust is only contained in Epinions dataset and the other two are in both datasets):

1. Trust information among users: if one user trusts another user, the latter one will be treated as the former one's neighbor.
2. Similarity of users (please refer to [96] for definition): if the similarity between two users is larger than a threshold, an edge is connected between them denoting they are neighbors of each other. Referring to [96], we set the value of this threshold 0.4 for movieLens dataset and 0.2 for Epinions dataset.

3. Similarity of items (please refer to [96] for definition): if the similarity between two items is larger than a threshold, an edge is connected between them denoting they are neighbors of each other. Referring to [96], we set the value of this threshold 0.4 for movieLens dataset and 0.2 for Epinions dataset.

3.4 Algorithms

In this section, we introduce the details of learning and inference processes of MCCRF.

3.4.1 Learning Process

Parameters learning is to obtain parameter $\{\alpha, \beta, \gamma\}$ which can maximize the log-likelihood from training data $D = \{(x_k, y_k)\}_{k=0}^{N}$, where x is observations and y is predictions. (x_k, y_k) is a training data sample, the setup of which will be explained in the experimental section. In this work, Gradient Ascent is chosen as optimization method. For simple denotation, we use vector λ to denote feature function weights $\{\alpha, \beta, \gamma\}$, and use vector $F(y_k, x_k)$ to denote the value of feature function vectors $\{H, G, R\}$ given y_k and x_k. Then, the log-likelihood can be written in

$$\begin{aligned} L_\lambda &= \sum_{k=0}^{N} \log p_\lambda(y_k|x_k) \\ &= \sum_{k}^{N} \left[\lambda \cdot F(y_k, x_k) - \log Z_\lambda(x_k)\right]. \end{aligned} \tag{3.12}$$

As discussed in [119], to make the integration Z calculable, we must have $\lambda > 0$. Thus it is substituted in algorithm by another variable in order to employ Gradient Ascent optimization

method. Let $\lambda = e^{\lambda'}$, where $e^{\lambda'}$ is set by $e_i^{\lambda'} = e^{\lambda'_i}$. Thus

$$L_\lambda = L'_{\lambda'} = \sum_k^N [e^{\lambda'} \cdot F(y_k, x_k) - \log Z_{e^{\lambda'}}(x_k)]. \tag{3.13}$$

The gradient of the objective function is

$$\nabla L'_{\lambda'} = e^{\lambda'} \cdot \sum_{k=0}^{N} \left[F(y_k, x_k) - E_{p_{\lambda'}(Y|x_k)} \left(F(Y, x_k) \right) \right]. \tag{3.14}$$

To calculate the expectation term is expensive. In this work, we propose an approximate estimation method based on Markov chain Monte Carlo. Particularly, we employ Gibbs sampling technique as our method. The main idea is to first sample a sequence of variables y following the distribution of current $p(y|x)$ (this distribution is defined in Eq. (3.7) and is decided by current λ). Then, the feature function values of the sequence data y are averaged as the expectation of feature function value denoted as

$$E_{p_\lambda(Y|x_k)}(F(Y|x_k)) = \frac{1}{S}(\sum_1^S F(\tilde{y}, x_k)), \tag{3.15}$$

where S is the length of the sequence.

One of the key points for Gibbs sampling is to calculate $p(y_{l,m}|y_{-l,-m}, X)$ in sampling the sequence, where $y_{-l,-m}$ denotes all other predictions except $y_{l,m}$. In our case,

$$P(y_{l,m}|y_{-l,-m}, X) = \frac{P(y_{l,m}, y_{-l,-m}|X)}{\int_{y_{l,m}} P(y_{l,m}, y_{-l,-m}|X) dy_{l,m}}. \tag{3.16}$$

Under the definition of $p(y|x)$ in Eq. (3.7), it is not difficult to conclude that $p(y_{l,m}|y_{-l,-m}, X)$ is a Gaussian distribution, the mean and variance of which can be calculated by current $y_{-l,-m}$, x and λ. Thus the Gibbs sampling methods is feasible in this estimation case by using existing Gaussian distribution

Algorithm 1 Learning Algorithm for MCCRF

Input: Training data $D = \{(x_k, y_k)\}_{k=0}^{N}$,
U: number of updating iterations
S: number of sampling iterations
Algorithm:

```
for i = 0 to N-1 do
    Load features
    Initialize λ, y
end for
Gibbs sampling initialization
for i = 0 to U-1 do
    for k = 0 to N-1 do
        for j = 0 to S-1 do
            for each user-item pair t in (x_k, y_k) do
                Sample y_t according to Eq. (3.7) and Eq. (3.16)
                Update distributions of y for relevant user-item pairs
            end for
        end for
    end for
    Compute the expectation term according to Eq. (3.15)
    Compute ∇λ' according to Eq. (3.14)
    Update λ'' = λ' + η * ∇λ'
end for
```

Output: Parameter λ of MCCRF model.

sampling methods (in this work, we use DistLib[1]) as tools. Due to space limitation, please refer to [7, 90] for more details about the theory of Gibbs sampling. The detailed learning algorithm is shown in Algorithm 1.

3.4.2 Inference Process

Inference is to search predictions that can maximize the joint probability density function conditioned on observations, which is formulated as

$$\hat{y} = \arg\max p(y|x).$$

[1] http://statdistlib.sourceforge.net

On this problem of MCCRF, exact estimation is hard to calculate, thus we still consider approximate methods. Generally speaking, Gibbs sampling can be directly used to estimate the optimal solution, however, as discussed in [7], this method is inefficient because random samples can rarely approach the optimal solution unless $p(y|x)$ has large probability mass around the solution. Thus, in this work, we employ Simulated Annealing. Using this strategy, the joint conditioned probability function of acceptable sampling data sequence can be controlled by the temperature schema as

$$p_i(\tilde{y}|x) = p^{1/T(i)}(\tilde{y}|x), \tag{3.17}$$

where $T(i)$ is the temperature at time i. When temperature falls, probability mass around the optimal solution will increase, making the sampling process approach to the solution faster. More details about simulated annealing in MCMC are shown in [7, 39, 90].

Utilizing MCMC technique as inference method has another advantage: it is easy to add constraints in the inference process to improve the prediction results. In relational recommendations, users usually have rating history on some items, and these ratings can serve as constraints in the inference to assist predictions. In our proposed framework, the constraints can be added into the model by fixing the rated scores in the inference process when sampling. Referring to [60, 78, 87], such process will not destroy the Markov property of the Conditional Random Fields model, and the inference result will be the best one in candidates that can fit the constraints. The detailed algorithm for inference of MCCRF is shown in Algorithm 2.

Algorithm 2 Inference Algorithm for MCCRF

Input: Testing Data
T_i: time control sequence
S: number of sampling iterations
λ: function weights vector
Algorithm:

```
Load features, λ, constraints
Fix predictions of relevant user-item pairs
Initialize predictions
Gibbs sampling initialization
for T = T_0 to T_min according to T_i do
  for i = 0 to S-1 do
    for each user-item pair t do
      if (prediction is not fixed by constraints) then
        Sample y_t according to Eq. (3.7), Eq. (3.16) and Eq. (3.17)
        Calculate ΔF defined in Simulated Annealing
        if (min(1, exp(−ΔF/T)) > random[0, 1]) then
          Accept y_t
          Update relevant distributions
        end if
      end if
    end for
  end for
end for
```

Output: Predictions of MCCRF.

3.5 Experiments

Our experiments are conducted on two real world datasets from MovieLens and Epinions. We aim at verifying the following issues:

1. How about the overall performance of our proposed approach comparing with traditional CF methods?

2. How does the relational dependency in predictions affect the accuracy of recommendation results?

3. How do the features we combined from previous work affect

the recommendation results?

4. How about the computing complexity of MCCRF?

To Issue 1, we compare our approach with traditional CF algorithms in Section 3.5.4; to Issues 2 and 3, additional experiments are conducted to show the effectiveness of relational dependency and combination of various features in Section 3.5.5 and Section 3.5.6. We give analysis of Issue 4 in Section 3.5.7. Experiments setup is introduced in Section 3.5.1, Section 3.5.2 and Section 3.5.3. In the pre-processing, clustering algorithms are employed, and the impact of cluster size is analyzed in Section 3.5.8.

3.5.1 Datasets

In this work, we choose two datasets, MovieLens[2] and Epinions[3] in our experiments for relational recommendation. MovieLens is a famous dataset in CF tasks. In this dataset, there are 1,682 movies and 943 users. Ratings are given on the scale of 1 to 5, with higher value indicating better satisfaction. There are totally 100,000 rating records in this user-item matrix. The density is

$$\frac{100,000}{1,682 * 943} = 6.3\%.$$

For a single user, there are at least 20 ratings. Some of the statistical results are shown in Table 3.1. Besides rating information, the dataset also provides other content information. For a movie item, content information includes released date, genre, etc; and for a user, age, gender, occupation are provided. In our approach, genre, occupation, age and gender are combined as content features.

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

Table 3.1: Statistics of MovieLens and Epinions

Statistics	MovieLens	Epinions
Min. Num. of Ratings/User	20	1
Min. Num. of Ratings/Item	1	1
Max. Num. of Ratings/User	737	1022
Max. Num. of Ratings/Item	583	2018
Avg. Num. of Ratings/User	106.04	16.55
Avg. Num. of Ratings/Item	59.45	4.76

Epinions dataset comes from a consumer review site Epinion.com. In this system, users can give reviews (scale from 1 to 5) to products, being used for future customers as reference and for companies to receive feedbacks or to recommend items. Different from traditional benchmark datasets, Epinions dataset has social trust information among users besides basic rating records. A user can build a trust/distrust list of other users for personalized products ranking as well as indicating users' reputations in the whole social network. Thus it is a good dataset for relational recommendation. The whole dataset contains 40,163 users who rated a total number of 139,529 different items at least once, writing 664,824 reviews. The density is

$$\frac{664,824}{40,163 * 139,529} = 0.01186\%.$$

There are totally $487,183$ trust information records in our dataset. The density of trust relationship is

$$\frac{487,183}{2 * C^2_{40,163}} = 0.0302\%.$$

Other statistics are summarized in Table 3.1.

In both datasets, we randomly group users into four groups, with three groups as training, and the rest as testing. To observe the performances when active users have different number of ratings as history, experiments are conducted by selecting 5,

10 and 15 as rating history for each active user respectively in MovieLens and 2, 5, and 10 in Epinions. We name them Given2, Given5, Given10, and Given15.

3.5.2 Data Sample Building

In this section, we introduce how we build probabilistic graphs on the two datasets. A probabilistic graph represents a data sample (x_k, y_k) in dataset $D = (x_k, y_k)_{k=1}^{N}$. For MovieLens, since it is small in size, all users and items can be contained in one probabilistic graph. For Epinions, the size is large. For this problem in memory-based CF, Xue et al. [163] proposed a cluster-based method as a solution. By clustering users into small groups, non-similar users are removed in predicting a particular user's evaluations. Thus not only the scalable problem is solved, the accuracy can also be improved. In this work, we employ similar ideas in our approach. Both users and items are clustered into sub-groups, and a probabilistic graph is built on one group of users and one group of items. Referring to [163], we employ K-means algorithm as our clustering algorithm. K is the number of clusters, which is manually defined. In this algorithm, we first randomly select K nodes (users/items) as centroid. All other nodes are assigned into a cluster whose centroid is closest to current node. During iteration processes, the centroid of each cluster is re-calculated based on current nodes in the cluster, and then other nodes are re-assigned to adapt the new centroid configuration. In each iteration, the node which has the smallest average distance to other nodes are selected as centroid. Similar to [163], we employ PCC to measure the

distance between two nodes. For users, it is defined as

$$Sim(a,u) = \frac{\sum_{i \in I(a) \cap I(u)} (r_{a,i} - \overline{r}_a)(r_{u,i} - \overline{r}_u)}{\sqrt{\sum_{i \in I(a) \cap I(u)} (r_{a,i} - \overline{r}_a)^2} \sqrt{\sum_{i \in I(a) \cap I(u)} (r_{u,i} - \overline{r}_u)^2}}, \quad (3.18)$$

where a and u denote two users. $I(a)$ and $I(u)$ are the items they have rated. $r_{a,i}$ is the rating of item i by user a. $\overline{r}_a$ is the average rating of user a. For items, the definition is similar. Due to space limitation, please refer [96] for the details of the definition. In Section 3.5.8, we will give analysis on the impact of cluster size K in this task.

3.5.3 Metrics

We use Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) as our evaluation metrics. MAE is defined as

$$MAE = \frac{\sum |R_{u,i} - \tilde{R}_{u,i}|}{N}, \quad (3.19)$$

where $\tilde{R}_{u,i}$ is the predicted ratings of item i by user u, $R_{u,i}$ is the ground truth, and N is the total number of testing predictions. RMSE is defined as

$$RMSE = \sqrt{\frac{\sum (R_{u,i} - \tilde{R}_{u,i})^2}{N}}. \quad (3.20)$$

In both metrics, lower value indicates higher accuracy.

3.5.4 Overall Performance

To compare our approach with traditional methods, we choose two algorithms (one memory-based and one model-based) as baselines. In memory-based methods, user-based PCC [17] and

Table 3.2: Performance in MovieLens dataset

Methods	MAE			RMSE		
	Given5	**Given10**	**Given15**	**Given5**	**Given10**	**Given15**
EPCC	0.835	0.830	0.815	1.065	1.059	1.033
AM	0.827	0.819	0.816	1.041	1.031	1.025
Fusion	0.815	0.806	0.805	1.029	1.024	1.022
EMDP	0.811	0.804	0.801	1.036	1.019	1.020
MCCRF	**0.784**	**0.781**	**0.778**	**0.995**	**0.994**	**0.988**

Table 3.3: Performance in Epinions dataset

Methods	MAE			RMSE		
	Given2	**Given5**	**Given10**	**Given2**	**Given5**	**Given10**
EPCC	0.887	0.867	0.858	1.136	1.105	1.092
AM	0.893	0.885	0.863	1.132	1.131	1.101
Fusion	0.885	0.860	0.853	1.132	1.092	1.101
EMDP	0.885	0.861	0.857	1.131	1.094	1.091
MCCRF	**0.871**	**0.845**	**0.837**	**1.115**	**1.078**	**1.067**

item-based PCC [130] are widely used. In our baseline, following the idea in [96] which improves the accuracy, we linearly combine these two methods, denoted as EPCC. For model-based methods, generative models are respective. Specifically, Aspect Model (AM) [67] is chosen as baseline. Since our approach belongs to memory-based methods, we choose two state-of-the-art memory-based methods, Similarity Fusion (Fusion) [157] and EMDP [96], for comparison. As stated before, these methods tried to solve similar problems with our approach, but our model have more advantages for solving the error propagation problem.

Table 3.2 and Table 3.3 shows the overall performance of different methods on MovieLens and Epinions, respectively. Lower MAE and RMSE values indicate better accuracy. On both datasets, we can conclude that MCCRF outperforms traditional and state-of-the-art algorithms. We summarize the improvements from two factors: relational dependency within predictions and combination of various features.

3.5.5 Effectiveness of Relational Dependency

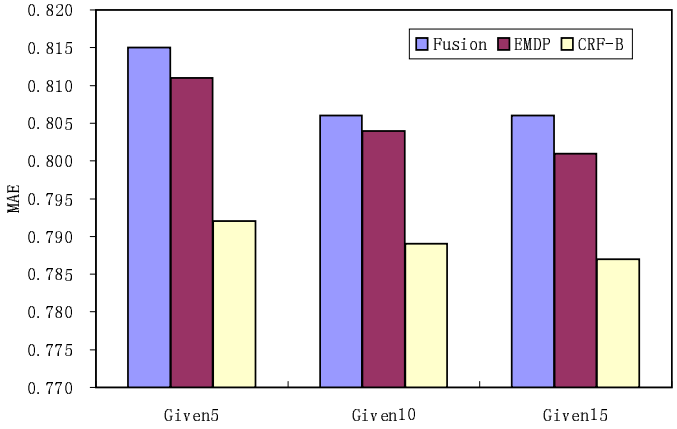
To evaluate the effectiveness of relational dependency in predictions, we conduct experiments with only basic features (CRF-B) of user/item similarities. This means we use the same information comparing with previous work, and the main difference of our approach is that we add relational dependency in predictions. The two state-of-the-art memory-based methods, Fusion method and EMDP method, are chosen for comparisons. Figure 3.4 and Figure 3.5 show the experimental results on the two datasets.

From these two figures we can conclude that relational dependency within predictions can improve recommendation results. This is because predictions of user-item pairs can help each other without error propagation. As the data is very sparse in real recommendation systems, utilizing relations in social network sufficiently can improve the accuracy.

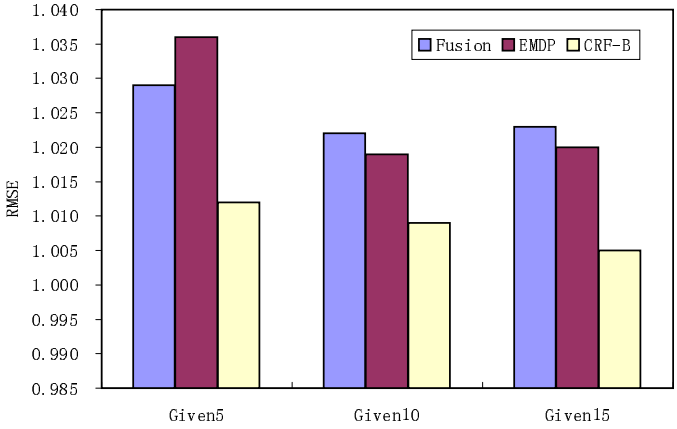
3.5.6 Effectiveness of Various Features

To evaluate the effectiveness of various features, we conduct experiments by adding features separately to basic features of user/item similarity. In MovieLens, we conduct experiments by adding occupation features (CRF-BO), age and gender features (CRF-BA), and genre features (CRF-BG). We compare the results with only basic features (CRF-B) and all features (CRF-All). In Epinions, we compare models with (CRF-T) and without (CRF-B) trust information. Figure 3.6 and Figure 3.7 show the results in the two datasets.

We can observe that each feature we combined (CRF-BO, CRF-BA, CRF-BG, CRF-T) can improve the prediction accuracy comparing to CRF-B. The combination of all features (CRF-ALL, CRF-T) can outperform models with single additional feature.

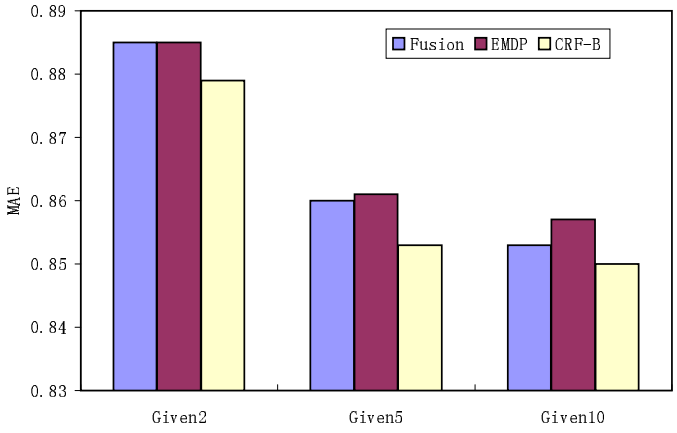


(a) MAE

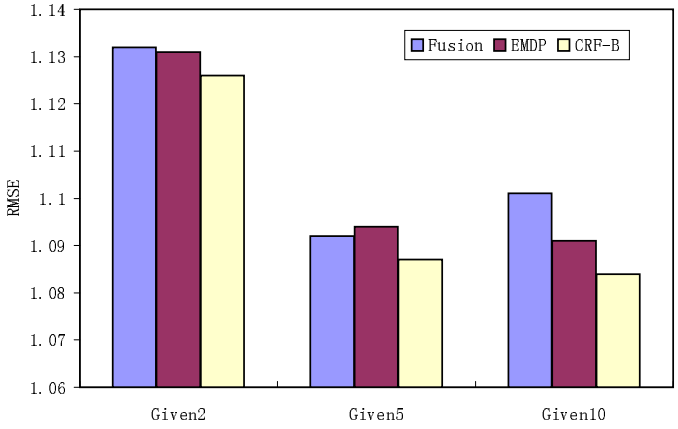


(b) RMSE

Figure 3.4: Effectiveness verification of the dependency features in MovieLens

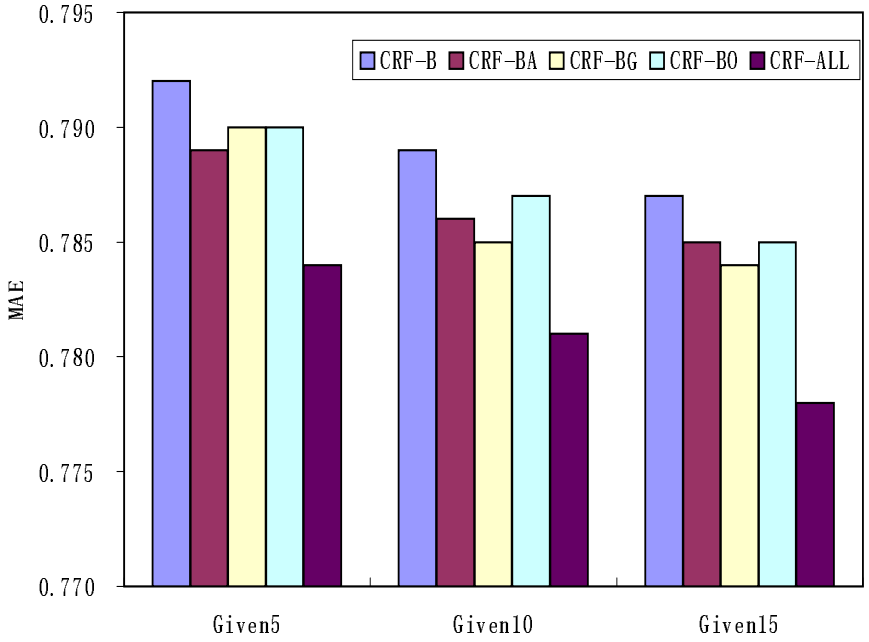


(a) MAE

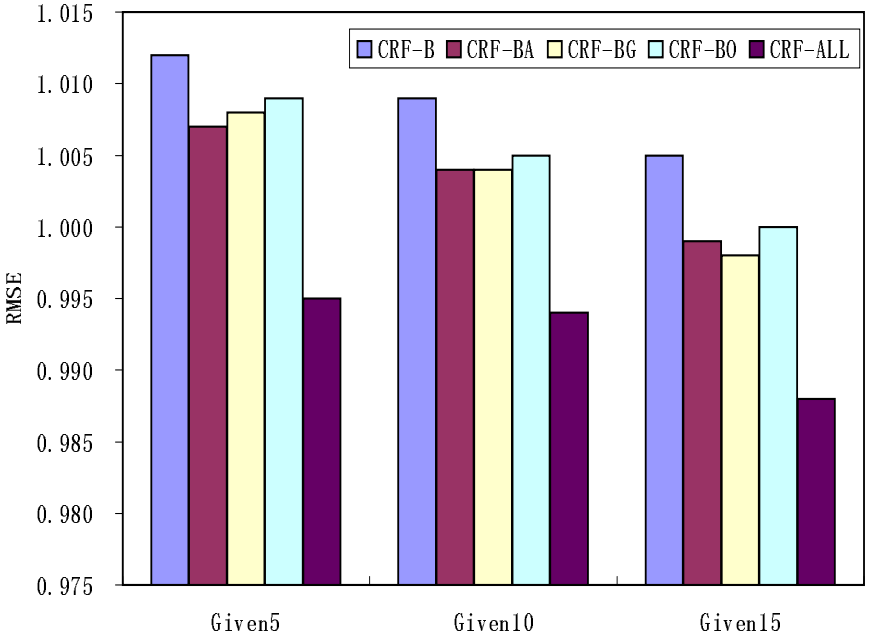


(b) RMSE

Figure 3.5: Effectiveness verification of the dependency features in Epinions

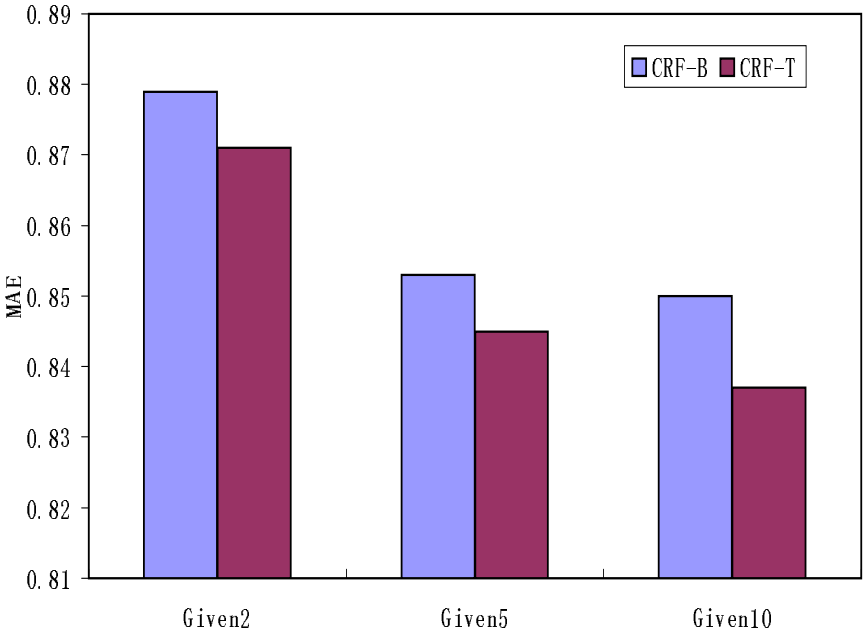


(a) MAE in MovieLens

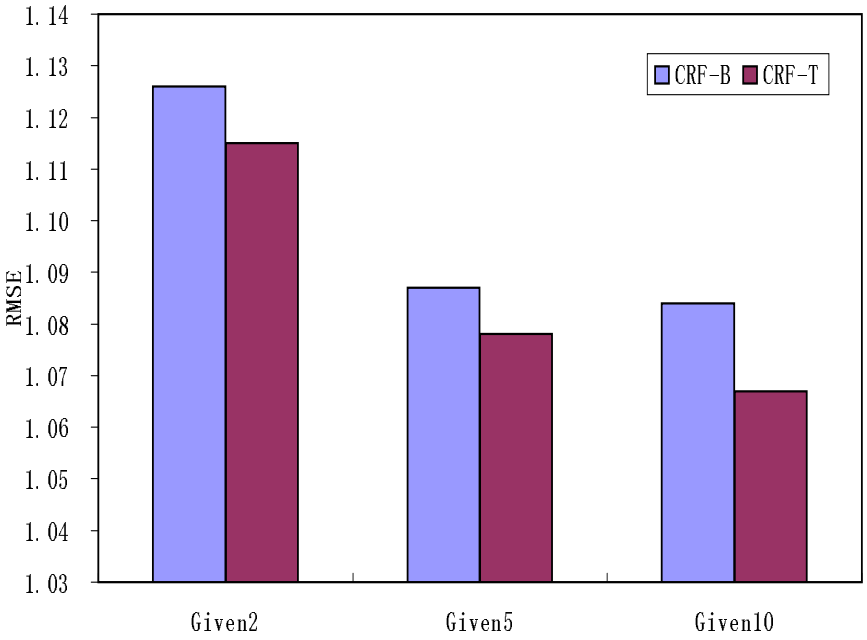


(b) RMSE in MovieLens

Figure 3.6: Effectiveness verification of local features



(a) MAE in Epinions



(b) RMSE in Epinions

Figure 3.7: Effectiveness verification of relational features

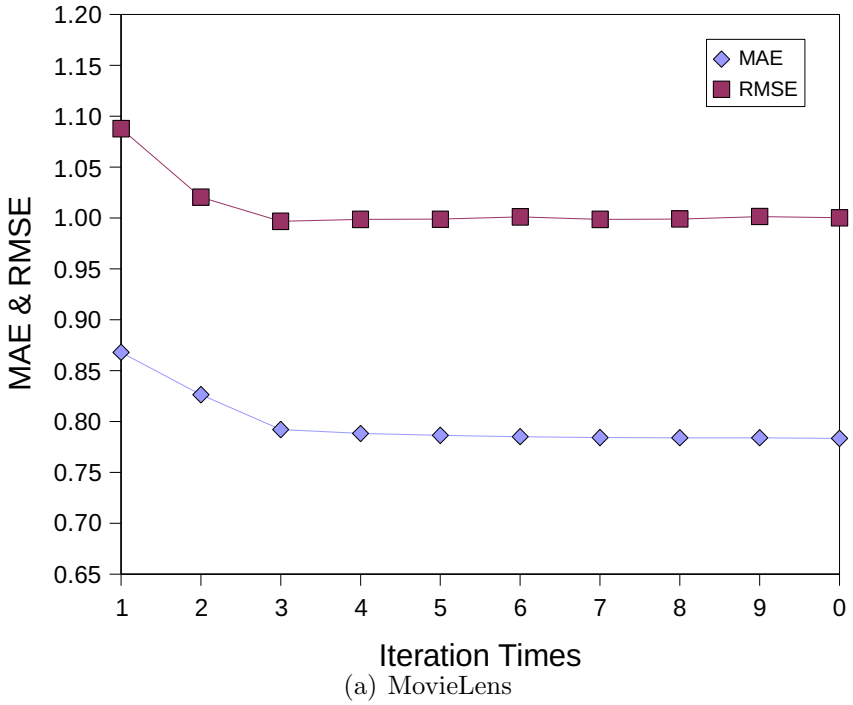
3.5.7 Computing Complexity Analysis

The main computation in our model lies in the sampling process in both training and inferencing. The number of sampling times is the key factor. It is determined by the number of sampling iterations at each temperature and the temperature control schema. Figure 3.8 shows the results of different iterations in the initialized temperature on two datasets. We can observe after four iterations, the change is not obvious. Figure 3.9 and Figure 3.10 show the results in different temperatures. According to these results, we set iteration number be 4 and temperature schema from 1.0 to 0.2 with interval of 0.2. Suppose there are m items and n users, the sampling times is $O(m * n)$. Another computation comes from the the updating process of Gaussian distributions of user-item pairs. This is decided by the neighbor size of current user-item pair. The neighbor size s can be controlled by adjusting the threshold mentioned in Section 3.3.4. The updating times for each sample of user-item pair is $O(s)$.

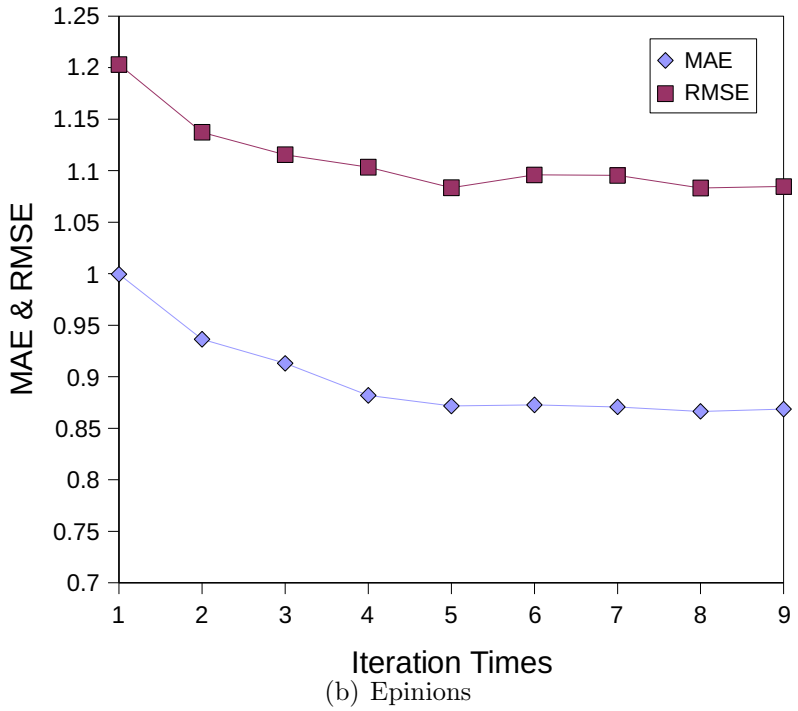
In our experiments, the testing hardware environment is on two Windows workstations with four dual-core 2.5GHz CPU and 8GB physical memory each. The approximate total time for inference in Epinions dataset is 9 hours.

3.5.8 Impact of Cluster Size

As discussed before, we employ clustering techniques as preprocessing. We conduct experiments on different settings to see the impact of cluster size. Figure 3.11 shows the experimental results (x-axis: userSize*itemSize). The accuracy increases first and then falls down. This is because at the beginning, there are not enough reference resources. But as the size of a cluster enlarges, non-relevant users/items are included, which influences the accuracy. In our experiments, items are clustered into 50 groups and users are clustered into 20 groups.

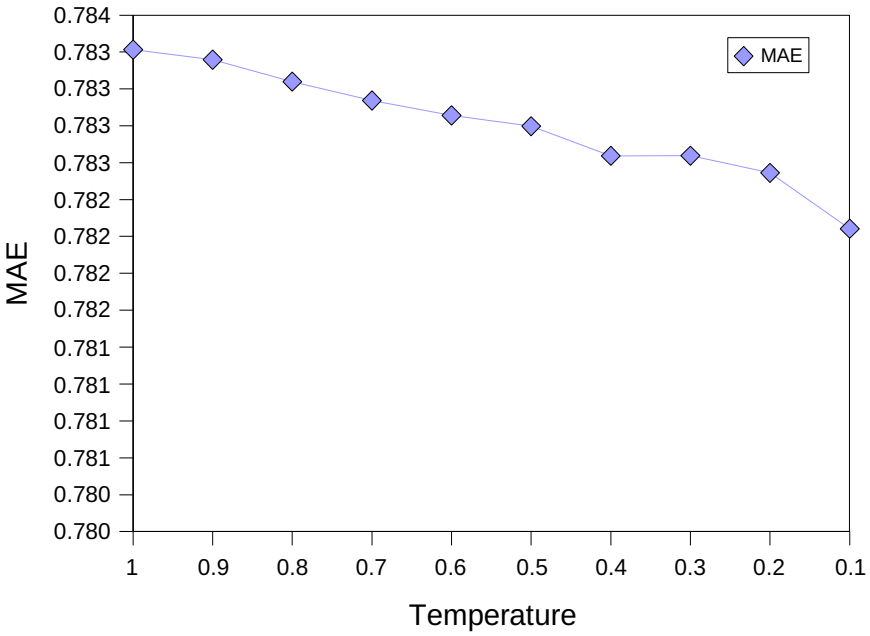


(a) MovieLens

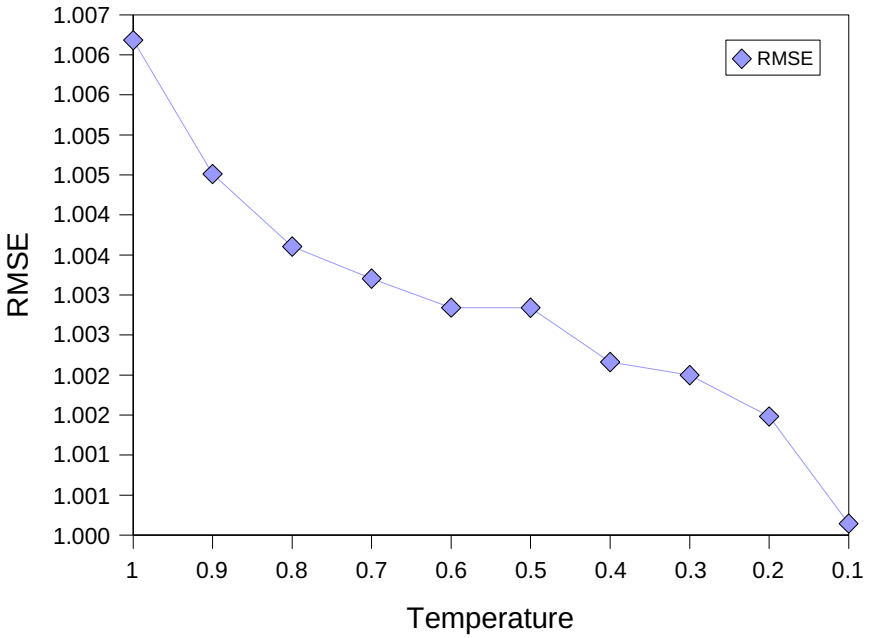


(b) Epinions

Figure 3.8: Result samples in different iteration times

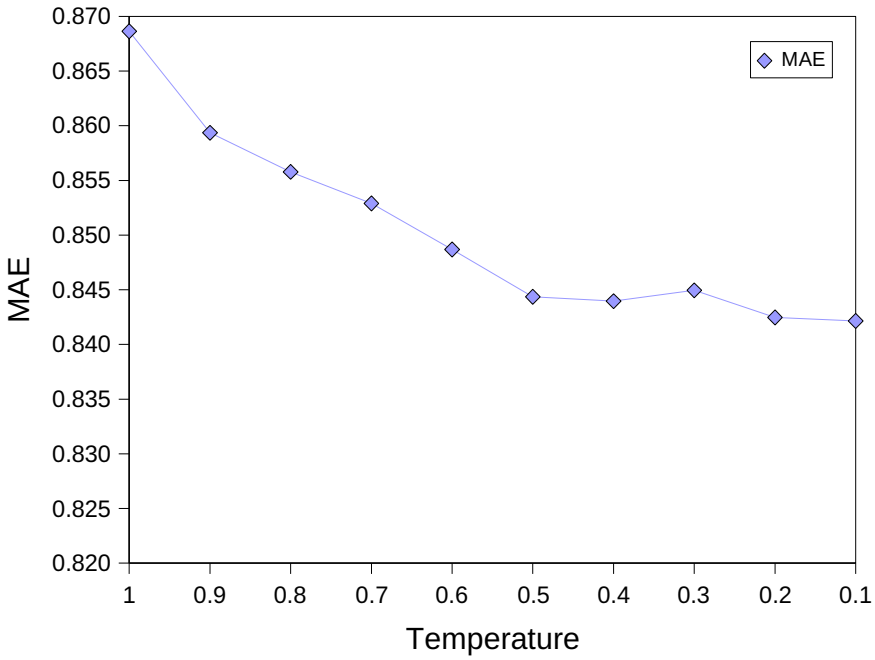


(a) MAE in MovieLens

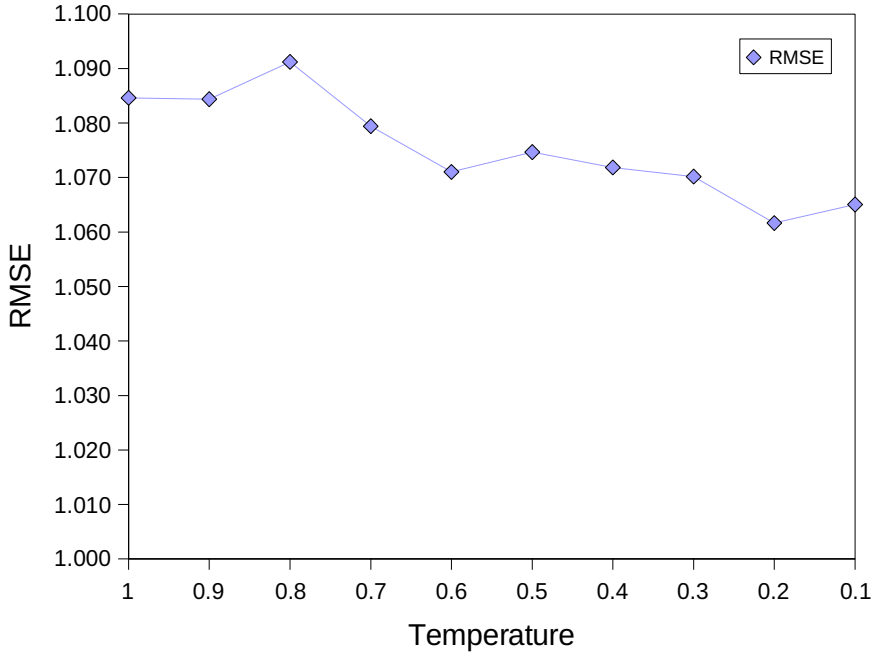


(b) RMSE in MovieLens

Figure 3.9: Result samples in different temperatures in MovieLens

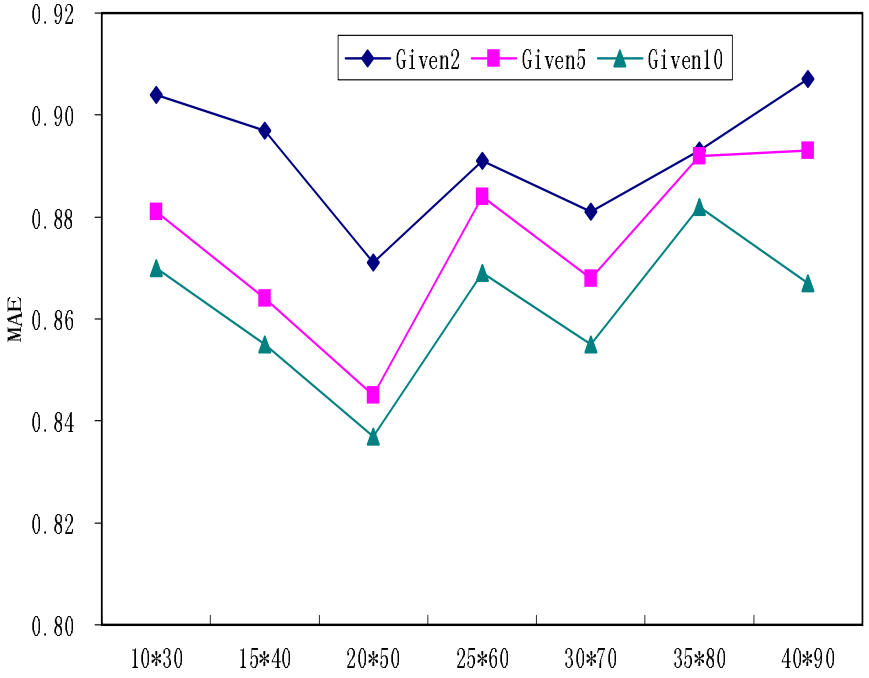


(a) MAE in Epinions

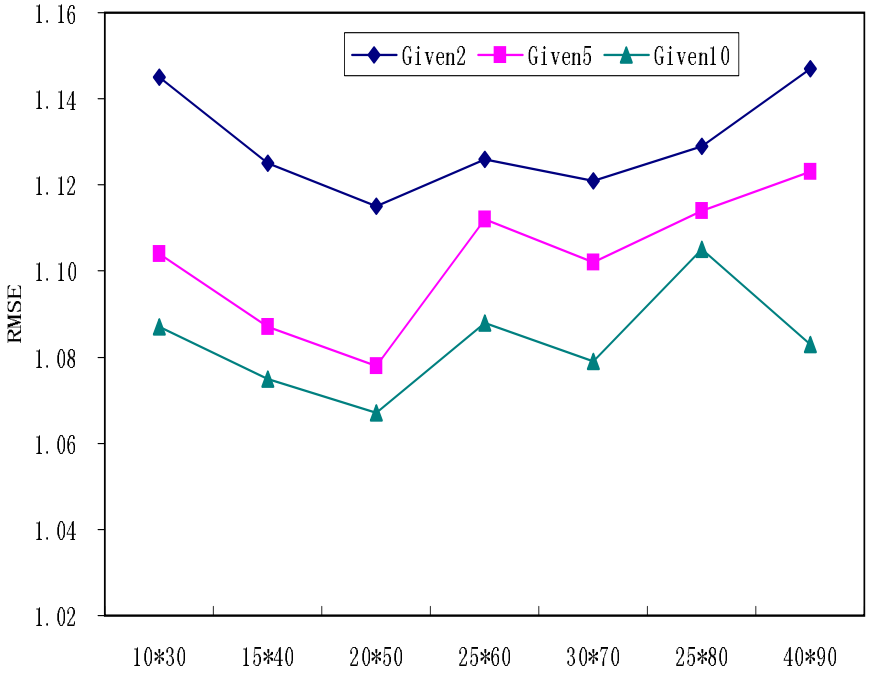


(b) RMSE in Epinions

Figure 3.10: Result samples in different temperatures in Epinions



(a) MAE



(b) RMSE

Figure 3.11: Results for different cluster sizes in Epinions

3.6 Summary

In this work, we have investigated relational fusion technique of multiple features. According to limitations of traditional fusion algorithms, we extend single-scale continuous CRF in theory and propose a new model multi-scale continuous CRF as an effective approach for relational fusion. We also propose MCMC-based methods for training and inference of the model. Experimental results on real world datasets, MovieLens and Epinions, have demonstrated: (1) Markov property in our approach is an effective technique to model the relational dependency within predictions. In sparse data, utilizing this kind of dependency can improve recommendation results. (2) The model is effective to combine various features.

Chapter 4

Effective Fusion of Regression and Ranking for Multi-measure Adaption

4.1 Limitation of Single-measure Adaption

Collaborative filtering (CF) are mainly divided into regression-oriented and ranking-oriented algorithms. Regression CF is a classical research topic. The goal is to generalize an estimation function to predict the ratings of each item for each user. Thus the rating information is modeled and the performance is evaluated on the closeness of predicted ratings and the user-given ratings. The main metrics used include Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). In applications, items with higher predicted ratings will be recommended to users. Ranking-oriented CF is proposed recently [91]. The goal is to predict the ranking relation of items for each user directly, the rating prediction is not necessary [91]. Thus the ranking relation information is directly modeled and the performance is evaluated on the ranking order of predicted items. The main metric used is the Normalized Discount Cumulated Gain (NDCG). Currently, each kind of algorithms is further divided into model-based [128, 139] and memory-based algo-

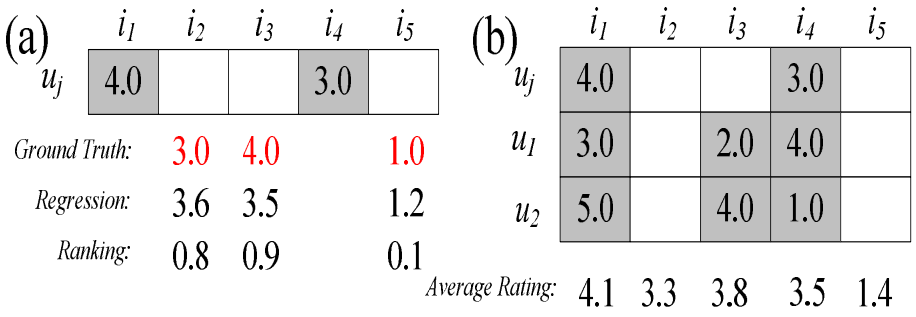


Figure 4.1: Examples to show the limitations of single-measure collaborative filtering algorithms

rithms [17, 91].

Through the decades, the collaborative filtering techniques have been explored deeply and have achieved significant improvement; however, regression-oriented and ranking-oriented algorithms are separately investigated, which has the following limitations: (1) over-bias limitation in single criterion and (2) information not fully utilized for the data sparsity problem.

Over-bias limitation in single measure. The over-bias here means that in some cases, regression-oriented algorithms cannot adapt to ranking and ranking-oriented algorithms cannot adapt to regression. Thus the performance will be influenced negatively. Take Fig. 4.1 (a) as an example. There are totally five items (i_1 to i_5), and one user u_j. i_1 and i_4 (gray color) are users' rating history and i_2, i_3 and i_5 (white color) are the items for recommendation. The first line shows the ground truth, the second line shows the prediction of a regression algorithm, and the third line shows the prediction of a ranking algorithm. For regression algorithms, although the predictions are close to the ground truth, it cannot guarantee the order of items. In this example, it changes the rank order of i_2 and i_3. In this case, if only recommending one item, the top-ranked item i_3 will be ignored. For ranking algorithms, it keeps the ranking order of the three items, but the predicted score may ignore the rating

information. In this example, the results are probabilities in the range of $[0, 1]$. In this case, if an user requests three items, i_5 may have the chance to be recommended together with i_2 and i_3. However, as indicated from the regression, i_5 is a very low-rated item and should not be recommended. Thus ranking algorithms fail to identify these items in recommendation. From the above analysis, the over-bias in single criterion will limit the performance of recommendation in real applications.

Information not fully utilized for data sparse problem. In the application of recommender system, data sparsity is a classical problem [96]. In the dataset of MovieLens[1], the density of existing ratings in the user-item matrix is 6.3%, and in Netflix[2], the density is 1.18%. In many cases, the data is too sparse for accurate recommendation. If the rating information and ranking information are separately modeled, relevant information will not be fully utilized. Regression algorithms ignore the ranking information and ranking algorithms ignore the rating information. But both kinds of information are very useful. Take an example shown Fig. 4.1 (b), where the previous user u_j has only two ratings. For regression algorithms, user u_1 will be seen as more similar to user u_j because the ratings of u_1 are closer to those of u_j than u_2 to u_j. However, u_2's rating for i_3 is closer to the ground truth in Fig. 4.1 (a). The reason is that in the training of regression algorithms, it fails to consider that u_1 changes the order of i_1 and i_4. If there is penalty for this, the performance would be improved. Thus the ranking information would be useful. For ranking algorithms, in this example, it has no information to predict the ranking of i_2 and i_5. However, the average rating information of i_2 and i_5, on the assumption that it is available, can be considered to help in ranking predication. Thus rating information would be also useful. Therefore,

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

if such information is not fully utilized, the algorithms will have limitations in solving the data sparsity problem.

4.2 Fusion Tasks for Multi-measure Adaption

From the above analysis, to solve the limitations, it is natural to combine the regression and ranking in collaborative filtering. Ideally, a good algorithm should perform well in both regression and ranking.

Intuitively, the combination is feasible. The objectives of two kinds of algorithms are mutually inclusive and there is no conflict for the combination in each criterion. Moreover, one objective function can also acts as the regularization for the other objective function. Thus they are expected to help each other and the combination is expected to enhance the performance.

In other domains such as documents/advertisements search application, previous work [133] has also demonstrated that the combination can enhance both performance. However, the methods in [133] cannot be utilized directly in CF because their methods are based on effective content features under the Support Vector Machine framework. In CF, content features are difficult to obtain for users with only a few ratings, and the prediction is based on the common behavior patterns of users. Thus new combination methods should be explored to solve this problem in CF. To the best of our knowledge, our work is the first attempt to combine the regression and ranking algorithms in CF applications.

Three questions arise in the investigation of combining regression and ranking in CF: 1) How to combine regression and ranking in CF? 2) Whether the combination will enhance the performance? 3) To what extent does the combination enhance? To answer these questions, we propose various combination methods in both model-based and memory-based CF algorithms.

Through experimental verification on two real-world datasets, MovieLens and Netflix, we demonstrate that the combinations are effective in improving the performance.

4.3 A Brief Review for Regression and Ranking Adaption in Recommender Systems

Regression-oriented CF and metrics. Regression-oriented recommendations are classical tasks in the past years. The goal is to predict the ratings as close as possible to the real ratings from the users. Then the items with high predicted ratings are selected as recommended results. Thus the metric is to mainly measure the closeness of the predicted and original ratings. Typical metrics include MAE and RMSE. MAE is defined as

$$MAE = \frac{\sum |R_{u,i} - \tilde{R}_{u,i}|}{N}, \tag{4.1}$$

where $\tilde{R}_{u,i}$ is the predicted ratings of item i by user u, $R_{u,i}$ is the ground truth, and N is the total number of testing predictions. RMSE is defined as

$$RMSE = \sqrt{\frac{\sum (R_{u,i} - \tilde{R}_{u,i})^2}{N}}. \tag{4.2}$$

In both metrics, lower value indicates higher accuracy. Since the problem has a long history, a large number of algorithms are developed for it. Some typical and competitive algorithms include [17, 65, 76, 77, 128, 158, 166, 170].

Ranking-oriented CF and metrics. Ranking-oriented recommendations are recently proposed tasks. The goal is to directly model and predict the ranking of unrated items without the intermediate step of rating predictions [91]. Thus the metric is to measure the closeness predicted ranking with the ground truth ranking. Typical metric is the NDCG value. Given the

rank of recommended results, NDCG at position P is defined as (referring to [91])

$$NDCG_{P-quality} = \frac{1}{U}\sum_{u}^{U} Z_u \sum_{p=1}^{P} \frac{2^{r_{u,p}} - 1}{\log(1+p)}, \tag{4.3}$$

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. For NDCG, a larger value indicates the higher accuracy. Comparing to regression algorithms, there is much less ranking algorithms. Some typical and competitive algorithms include [91, 92, 139].

Selected Methods for Combination. As discussed in the introduction section, both regression and ranking algorithms have limitations, we investigate the combination problem. Specifically, we propose combination methods for both model-based and memory-based algorithms. In choosing the fundamental components to combine with, we select typical and competitive algorithms in previous work. Without loss of generality, for model-based algorithms, we select probabilistic matrix factorization (PMF) [128] as regression method and list-wise matrix factorization (LMF) [139] as ranking method; and for memory-based algorithms, we select user-based Pearson correlation coefficient (PCC) [17] as regression method and EigenRank [91] as ranking method. All of them have been thoughtfully tested in previous work.

4.4 Effective Fusion of Regression and Ranking in Model-based Collaborative Filtering

In this section, we will introduce the combination of regression and ranking in model-based CF methods. The two com-

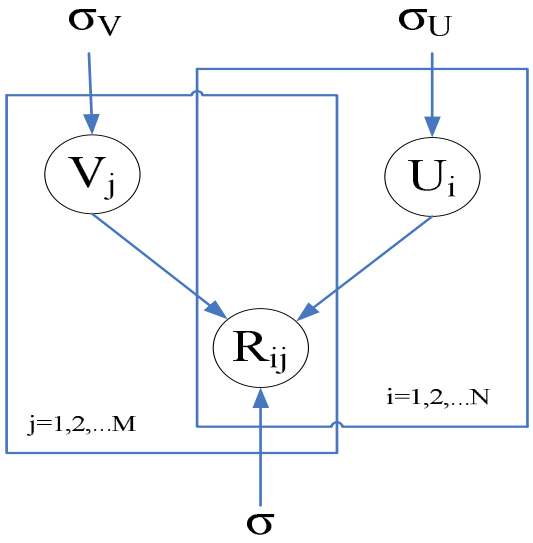


Figure 4.2: Probabilistic graph of probabilistic matrix factorization and list-wise matrix factorization

petitive algorithms selected are probabilistic matrix factorization (PMF) [128], which is regression-oriented; and list-wise matrix factorization (LMF) [139], which is ranking-oriented. Since both models share the same probabilistic graph with different objective functions (loss functions), in this section, we will first introduce the preliminary knowledge about the probabilistic graph and notations which will be followed by separate introduction of the two algorithms. Then we discuss how we combine them together and finally give the complexity analysis.

4.4.1 Preliminary Knowledge

Suppose there are N users and M items. Fig. 4.2 shows the probabilistic graph of PMF and LMF. For each user and item, there is an l-dimensional latent feature vector. The feature vectors for users are denoted as $U \in R^{l*N}$ and the feature vectors for items are denoted as $V \in R^{l*M}$. Let R_{ij} denote the rating of item j given by user i. In this graph, the distribution of R_{ij} is

defined as [128]

$$P(R|U,V,\sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{M}[N(R_{ij}|g(U_i^T V_j),\sigma^2)]^{I_{ij}}, \tag{4.4}$$

where $N(x|u,\sigma^2)$ is a Gaussian distribution with the mean u and variance σ^2. $g(x)$ is the logistic function $g(x) = 1/(1+\exp(-x))$ to convert $U_i^T V_j$ to $[0,1]$ scale. The real ratings in applications are usually a positive integer (e.g., from 1 to 5). In this work, without loss of generality, we map the ratings from 1 to K to $[0,1]$ by using $f(x) = (x-1)/(K-1)$ refering to [128]. I_{ij} is an indicator to describe whether user i has rated item j. The zero-mean spherical Gaussian priors are further utilized to U and V as

$$P(U|\sigma_U^2) = \prod_{i=1}^{N} N(U_i|0,\sigma_U^2 I), \tag{4.5}$$

$$P(V|\sigma_U^2) = \prod_{j=1}^{M} N(V_j|0,\sigma_V^2 I). \tag{4.6}$$

4.4.2 Regression-adapted Component in Model-based Collaborative Filtering

We choose PMF as the regression-adapted component in model-based collaborative filtering. The objective function of PMF is to find U, V by minimizing the summation of regression loss and regularization as

$$\arg\min_{U,V} Loss_{reg}(U,V) + Regularization(U,V), \tag{4.7}$$

where the loss function is defined as

$$Loss_{reg}(U,V) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - g(U_i^T V_j))^2, \tag{4.8}$$

and the regularization term is defined as

$$Regularization(U, V) = \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2). \tag{4.9}$$

From the definition, it is observed that the loss function is to minimize the mean square error. This is a regression-oriented loss function. The range is $[0, +\infty]$. The optimal value is 0. The regularizations are Frobenius norms for both U and V. Under the assumption in this probabilistic graphical model, the objective function has also another interpretation, which is equal to maximizing the posterior log-likelyhood of $P(U, V|R, \sigma_U^2, \sigma_V^2, \sigma^2)$ [128].

4.4.3 Ranking-adapted Component in Model-based Collaborative Filtering

We choose LMF as the ranking-adapted component in model-based collaborative filtering. The objective function of LMF is to find U, V that can optimize

$$\arg\min_{U,V} Loss_{rank}(U, V) + Regularization(U, V), \tag{4.10}$$

where the loss function is defined by Kullback-Leibler (KL) divergence [79] as

$$Loss_{rank}(U, V) = \sum_{i=1}^{N}\sum_{j=1}^{M} P_{l_i}(R_{ij}) \log(\frac{P_{l_i}(R_{ij})}{(g(U_i^T V_j))}), \tag{4.11}$$

and the regularization term is defined the same as PMF above. Here the $P_{l_i}(R_{i,j})$ is the top one probability given user i's ranking list l_i (e.g., with K items), which is defined in the list-wise learning to rank framework [22, 93]. The definition is

$$P_{l_i}(R_{ij}) = \frac{\exp(R_{ij})}{\sum_{k=1}^{K} \exp(R_{ik})}. \tag{4.12}$$

If the rating of item j in the list l_i is larger, from the definition, there will be larger chance for j to rank higher among the K items in the list.

Different from that in PMF, the loss function in LMF is to directly optimize the ranking on the framework of list-wise learning to rank. The intuitive meaning is the Kullback-Leibler divergence of probability derived from original and predicted ratings. The range is $[0, +\infty]$. The optimization goal is 0. Comparing to traditional pair-wise learning to rank, the advantage of list-wise is to reduce much calculation complexity while retaining the key idea of rank modeling.

4.4.4 Effective Fusion Approach

From the above analysis, we can see that both objective functions contain two parts: loss function and regularization. Also, a good property is that their ranges are the same and the optimal value is 0 for both. Thus for the combination, it is natural to linearly combine the loss function while sharing the same regularization. In the combined regression and ranking in model-based CF, we propose the following combined objective function

$$\begin{aligned}\min_{U,V}\ \zeta = \min_{U,V}\ & \alpha_1 Loss_{Reg}(U,V) + \alpha_2 Loss_{Rank}(U,V) \\ & + Regularization(U,V) \qquad (4.13)\\ = \min_{U,V}\ & \alpha_1 * \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - g(U_i^T V_j))^2 \\ & + \alpha_2 * \sum_{i=1}^{N}\sum_{j=1}^{M} P_{l_i}(R_{ij}) \log(\frac{P_{l_i}(R_{ij})}{P_{l_i}(g(U_i^T V_j))}) \\ & + \frac{\lambda}{2}(\|U\|^2 + \|V\|^2).\end{aligned}$$

When $\alpha_1 = 1$ and $\alpha_2 = 0$, the objective function is reduced to PMF; when $\alpha_1 = 0$ and $\alpha_2 = 1$, the objective function is reduced

to LMF; when $\alpha_1 > 0$ and $\alpha_2 > 0$, the objective function is the combined objective function. The two objective functions share the same regularization term to avoid over-fitting problem.

In fact, for both single-criterion algorithms, the combination can also be seen as adding another kind of regularization term to the original model. For example, $Loss_{Rank}$ can be seen as a regularization term for PMF model. While PMF optimize the RMSE, it should simultaneously keep the right order. As discussed before, intuitively, $Loss_{Rank}$ has no conflict with $Loss_{Reg}$. In the same way, $Loss_{Reg}$ can be seen as a regularization term for LMF model.

A local minimum of the combined objective function can be calculated by gradient descent optimization. The gradient with respect to U and V is

$$\begin{aligned}\frac{\partial \zeta}{\partial V_j} = \quad & \alpha_1 \sum_{i=1}^{N} -I_{ij}(R_{ij} - g(U_i^T V_j)) g'(U_i^T V_j) U_i \\ + \quad & \alpha_2 \sum_{i=1}^{N} I_{ij} \frac{\exp(g(U_i^T V_j))}{\sum_{k=1}^{M} I_{ik} \exp(g(U_i^T V_j))} g'(U_i^T V_j) U_i \\ - \quad & \alpha_2 \sum_{i=1}^{N} I_{ij} \frac{\exp(R_{ij})}{\sum_{k=1}^{M} I_{ik} \exp(R_{ik})} g'(U_i^T V_j) U_i \\ + \quad & \lambda V_j \end{aligned} \tag{4.14}$$

Algorithm 3 Algorithm of Combined Regression and Ranking for model-based CF

Inputs: Ratings of test user and users
Outputs: Predicted ratings for unrated user-item pairs

```
Read all the rating information
Initialize U and V randomly
Calculate current objective function value by Eq. (4.13)
Initialize update rate to 1
while iteration is not over do
    Calculate the gradient according to Eq. (4.14) and Eq. (4.15)
    Update U and V according to current rate
    Update objective function value by Eq. (4.13)
    Update rate
end while
Write U, V, and calculate predicted values
```

$$\begin{aligned}\frac{\partial \zeta}{\partial U_i} = \quad & \alpha_1 \sum_{j=1}^{M} -I_{ij}(R_{ij} - g(U_i^T V_j))g'(U_i^T V_j)V_j \\ + \quad & \alpha_2 \sum_{j=1}^{M} I_{ij} \frac{\exp(g(U_i^T V_j))}{\sum_{k=1}^{M} I_{ik} \exp(g(U_i^T V_j))} g'(U_i^T V_j)V_j \\ - \quad & \alpha_2 \sum_{j=1}^{M} I_{ij} \frac{\exp(R_{ij})}{\sum_{k=1}^{M} I_{ik} \exp(R_{ik})} g'(U_i^T V_j)V_j \\ + \quad & \lambda U_i \end{aligned} \tag{4.15}$$

Details of the algorithm are described in Algorithm 3.

4.4.5 Complexity Analysis

The main calculation of the combined algorithm comes from the calculation of gradient of the objective function. In our case, the complexity is $O(\rho_R * l)$, where the ρ_R is the number of nonzero ratings in the user-item matrix. Since the data is very

sparse, ρ_R is extremely small comparing to the total size of user-item matrix. Thus the computational time of the combination method is linear with respect to the number of observations. Therefore, the algorithm is efficient in large scale applications.

4.5 Effective Fusion of Regression and Ranking in Memory-based Collaborative Filtering

In this section, we will introduce the combination of regression and ranking in memory-based CF methods. The two competitive algorithms selected are user-based PCC
method [17], which is regression-oriented, and EigenRank [91], which is ranking-oriented. In memory-based methods, the main idea is to find similar users and make the prediction according to these users. Thus the key problem becomes how to calculate the similarity between users and how to predict the final rating/ranking. Regression-oriented algorithms utilize regression-based similarity calculation while ranking-oriented algorithms utilize ranking-oriented similarity calculation. We will first introduce the two methods separately and then discuss how we combine them together. Afterwards, we give the complexity analysis.

4.5.1 Regression-adapted Component in Memory-based Collaborative Filtering

We choose User-based PCC as the regression-adapted component in memory-based collaborative filtering. User-based PCC methods utilize Pearson Correlation Coefficient as similarity calculation for finding similar users for current user. It is defined

as

$$Sim(a,u) = \frac{\sum_{i\in I(a)\cap I(u)}(r_{a,i}-\overline{r}_a)(r_{u,i}-\overline{r}_u)}{\sqrt{\sum_{i\in I(a)\cap I(u)}(r_{a,i}-\overline{r}_a)^2}\sqrt{\sum_{i\in I(a)\cap I(u)}(r_{u,i}-\overline{r}_u)^2}}, \quad (4.16)$$

where a and u denote two users. $I(a)$ and $I(u)$ are the items they have rated. $r_{a,i}$ is the rating of item i by user a. $\overline{r}_a$ is the average rating of user a.

PCC is regression-oriented and it balances different users' evaluation standard. For example, some users are more likely to give high ratings and some tend to give low ratings. The similarity is based on the difference of values rather than absolute values.

By using PCC similarity, we can find users with high similarity to the current user as its neighborhood. Then, the rating predictions are based on the following formula

$$f(u,i) = \overline{u} + \frac{\sum_{u_a\in S(u)} Sim(u_a,u)(r_{u_a,i}-\overline{u}_a)}{\sum_{u_a\in S(u)} Sim(u_a,u)}, \quad (4.17)$$

where $S(u)$ is the neighborhood of the current user u, $\overline{u}$ is the user's average rating score.

4.5.2 Ranking-adapted Component in Memory-based Collaborative Filtering

We choose EigenRank as the ranking-adapted component in memory-based collaborative filtering. In EigenRank, Kendall Rank Correlation Coefficient (KRCC) [100] is utilized as similarity. Its definition is

$$S_{u,v} = 1 - \frac{4 * \sum_{i,j\in I_u\cap I_v} I^-((r_{u,i}-r_{u,j})(r_{v,i}-r_{v,j}))}{|I_u\cap I_v|(|I_u\cap I_v|-1)}, \quad (4.18)$$

where $I^-(x)$ is an indicator function that equals to 1 if $x < 0$ and equals to 0 otherwise. KRCC is a ranking-oriented similarity that counts the number of the same pair-wise ranking between two users.

To simplify the calculation for pairwise ranking, the prediction of EigenRank model is based on random walk theory. A probabilistic transition matrix P is defined as

$$p_{ij} = p(j|i) = \frac{e^{\psi(j,i)}}{\sum_{j \in S} e^{\psi(j,i)}}, \tag{4.19}$$

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

$$\psi(i,j) = \frac{\sum_{v \in N_u^{i,j}} s_{u,v} \cdot (r_{v,i} - r_{v,j})}{\sum_{v \in N_u^{i,j}} s_{u,v}}. \tag{4.20}$$

In this equation, $N_u^{i,j}$ is the set of u's neighbors, and $s_{u,v}$ is KRCC similarity. A stationary distribution is employed to decide the preference score of an item, which is defined as

$$\pi = \pi * P, \tag{4.21}$$

where π is the stationary distribution vector to decide the final ranking order of all items.

4.5.3 Effective Fusion Approach

Different from model-based algorithms, which can linearly combine the objective functions of regression and ranking, in memory-based methods, usually there is no direct optimization goal. Heuristic rules are commonly employed as an approximation. Thus in the combination of memory-based methods, we consider to directly combine the results of regression and ranking algorithms for an effective achievement of the combination.

The challenge lies in that the results of regression and ranking are usually not in the same scale, thus the result values are

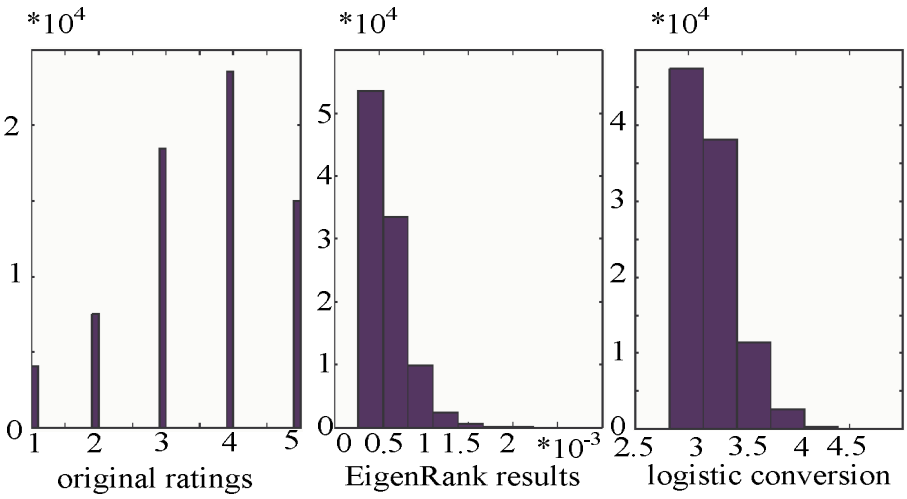


Figure 4.3: Problem illustration in data conversion

incompatible for linearly combination. For example, user-based PCC method returns a predicted rating for each item, which is from 1 to 5; EigenRank method, on the other hand, returns a stationary probability for each item, which is from 0 to 1. Another problem is that we need to evaluate the methods from both regression and ranking angles, thus the final prediction should have the value of predicted ratings. Therefore, we need to convert the result of EigenRank from the scale of $[0, 1]$ to $[1, 5]$. This is challenging because conversion from ranking to regression is more difficult than regression score to ranking as shown in previous work [47].

Some naive conversion methods can be applied intuitively. For example, directly multiply the results by 5. However, this can be problematic. Fig. 4.3 draws the distribution of existing ratings in MovieLens dataset (Left) and the distribution of EigenRank results (Middle). It can be seen that most results of EigenRank is around 0.001. Thus to multiply it by a constant will not work. Another intuitive method is to first convert the results using logistic function $1/(1.0 + \exp(-cx))$ and then to multiply it by a constant. But from the Fig. 4.3 (Right), it is

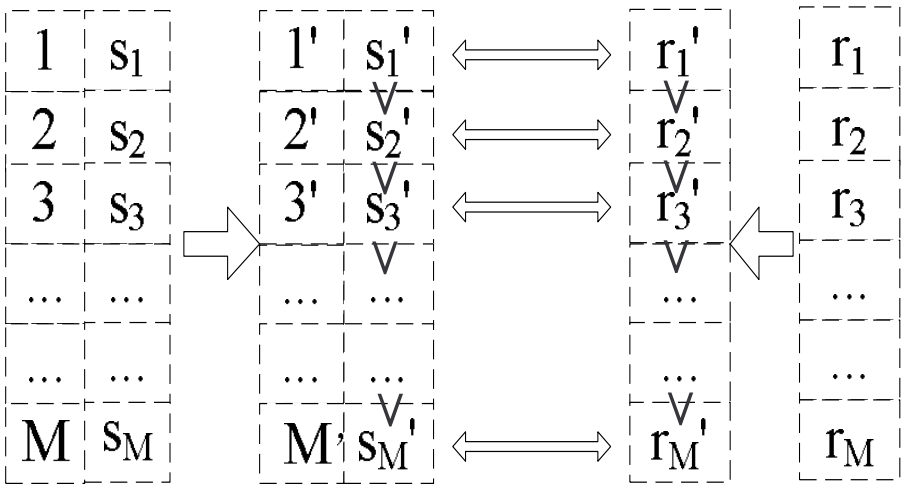


Figure 4.4: Illustration of the sampling trick

observed that the two distributions are quite different. Ratings follow Gaussian distribution while the converted scores follow more like power-law distribution. Thus it is still difficult to convert the value to a common rating distribution.

To solve this problem, we propose a sampling trick for the conversion purpose. Notice the goal is to generate a "normal" rating value for the EigenRank result, that has a reasonable distribution. Thus in the sampling trick, as shown in Fig. 4.4, we first sample M ratings $r_1, r_2, ..., r_M$ for unrated items of a user according to the distribution of all ratings in the training set. From Fig. 4.3 and many previous statistical work [96], the ratings follows a Gaussian distribution. Thus in the sampling, the mean and variance are learned from the training set. In the experiments, we also consider a user's average rating in evaluating the mean. Secondly, we rank both the EigenRank results and the sampled ratings in a decreasing order. Finally, we map the EigenRank results to the sampled ratings according to the order. In this way, the conversion guarantees that the results follow a reasonable distribution while keeping the ranking result from ranking-oriented algorithms.

Algorithm 4 Algorithm of Combined Regression and Ranking for memory-based CF

Inputs: Ratings of test user and users
Outputs: Predicted ratings for unrated user-item pairs

1: Read all the rating information
2: Find neighborhood of each user by PCC
3: Find neighborhood of each user by KRCC
4: **while** for each user **do**
5: Calculate regression result according to Eq. (4.17)
6: Calculate ranking result according to Eq. (4.21)
7: Combine the two results according to Eq. (4.22)
8: **end while**
9: Write the predicted value

Once the conversion is ready, we can linearly combine the results from regression-oriented and ranking-oriented algorithms as

$$Rate_{comb} = \alpha Rate_{reg} + (1 - \alpha) Rate_{rank}. \tag{4.22}$$

More details of the algorithm are shown in Algorithm 4.

4.5.4 Complexity Analysis

The main computation of our algorithm comes from two aspects: 1) similarity calculation; and 2) stationary distribution calculation from EigenRank.

For the similarity calculation of two users, in both PCC and KRCC, the complexity is $O(k)$, where k is the number of common items between the users. But we need to compare all the user pairs which is almost incomputable. In application, by following the idea in [128], both users and items are divided into small bins. This means it can find similar users in a limited number of user groups instead of all the users for approximation. Thus the complexity for finding similar users is $O(C * k * N)$, where C is constant and N is the number of users. It is linear with respect to the number of users.

Table 4.1: 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%

For the stationary distribution, as shown in [91], the complexity is $O(M)$ (M is the number of items) by utilizing the iterative power method. Therefore, the computation complexity also scales linearly with respect to the number of items and users.

4.6 Experiments

In this section, we will first introduce the datasets and relevant experimental setup. The experiments are conducted for two parts: combining regression and ranking in model-based algorithms, and that in memory-based algorithms. In each part, we are going to identify the following two issues: sensitivity analysis of combination parameters, and the overall performance in both regression-oriented and ranking-oriented measures.

4.6.1 Experimental Setup

We choose two datasets, MovieLens 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

Table 4.2: Performance of model-based combination in MovieLens

Methods	Given5				
	MAE	**RMSE**	**NDCG1**	**NDCG3**	**NDCG5**
PMF	0.870	1.153	0.640	0.663	0.673
LMF	0.967	1.350	**0.692**	0.687	0.701
RegPModel	**0.845**	**1.081**	0.672	0.676	0.692
RankPModel	0.863	1.132	0.681	**0.689**	**0.702**
Methods	**Given10**				
	MAE	**RMSE**	**NDCG1**	**NDCG3**	**NDCG5**
PMF	0.825	1.067	0.691	0.700	0.719
LMF	0.953	1.301	0.679	0.708	0.732
RegPModel	**0.809**	**1.003**	0.703	0.708	0.727
RankPModel	0.813	1.028	**0.705**	**0.719**	**0.736**
Methods	**Given15**				
	MAE	**RMSE**	**NDCG1**	**NDCG3**	**NDCG5**
PMF	0.780	0.97	0.693	0.730	0.753
LMF	0.946	1.290	0.734	0.748	0.763
RegPModel	**0.781**	**0.960**	0.742	0.750	**0.768**
RankPModel	0.786	0.964	**0.745**	**0.752**	0.765

value on the scale of 1 to 5, with a higher value indicating better satisfaction. More statistics are shown in Table 1.

In MovieLens, referring to the experimental setup in [163], we randomly choose 600 users for training and the remaining 343 users for testing. In Netflix, following the idea of [128], we randomly divide the users and items into small bins with around 1,000 users and 3,000 items each. The average is calculated as the final result. To observe the performance when the active users have different number of ratings as the history, experiments are conducted by selecting 5, 10 and 15 ratings as the rating history for each active user respectively in both datasets. We name them Given5, Given10, Given15. Users whose rating number is less than the configuration are not included in the evaluation. Before experiments, a pre-processing is conducted to rank all the ratings of a user in an ascending order according to the rating time stamp.

Table 4.3: Performance of model-based combination in Netflix

Methods	Given5				
	MAE	**RMSE**	**NDCG1**	**NDCG3**	**NDCG5**
PMF	0.820	1.069	0.694	0.691	0.691
LMF	0.979	1.404	0.713	0.702	0.715
RegPModel	**0.819**	**1.062**	0.697	0.685	0.696
RankPModel	0.828	1.063	**0.721**	**0.713**	**0.723**
Methods	**Given10**				
	MAE	**RMSE**	**NDCG1**	**NDCG3**	**NDCG5**
PMF	0.819	1.061	0.709	0.720	0.733
LMF	0.944	1.444	0.712	**0.721**	0.731
RegPModel	**0.781**	**0.979**	0.724	0.716	0.727
RankPModel	0.801	1.013	**0.732**	0.719	**0.735**
Methods	**Given15**				
	MAE	**RMSE**	**NDCG1**	**NDCG3**	**NDCG5**
PMF	0.769	0.947	0.749	0.742	0.763
LMF	0.918	1.292	0.722	0.743	0.764
RegPModel	**0.757**	**0.922**	0.747	0.750	0.722
RankPModel	0.762	0.931	**0.750**	**0.755**	**0.775**

For evaluation of the combination, we choose measures from both regression and ranking. The regression-oriented metrics we choose is MAE and RMSE, and the ranking-oriented metrics we choose is NDCG. The detailed definitions of these metrics have been discussed in previous section. For NDCG, we choose positions at 1, 3, and 5 (namely NDCG1, NDCG3, and NDCG5), which is practical in applications.

4.6.2 Performance in Model-based Fusion

Convergence

In model-based methods, the main optimization approach is gradient descent approach. Thus we analyze the convergence of the optimization. We show the convergence by the evaluation performance in each iteration in Fig. 4.5 (a) and (b) show the MAE and RMSE evaluation for MovienLens (a) and Netflix (b)

Table 4.4: Performance of memory-based combination in MovieLens

Methods	Given5				
	MAE	**RMSE**	**NDCG1**	**NDCG3**	**NDCG5**
PCC	0.877	1.257	0.668	0.671	0.690
EigenRank	0.878	1.287	0.684	0.709	0.719
RegPMemory	**0.817**	**1.099**	**0.696**	0.698	0.711
RankPMemory	0.848	1.194	0.693	**0.711**	**0.721**
Methods	**Given10**				
	MAE	**RMSE**	**NDCG1**	**NDCG3**	**NDCG5**
PCC	0.806	1.067	0.690	0.713	0.734
EigenRank	0.876	1.288	0.692	0.718	0.737
RegPMemory	**0.789**	**1.028**	0.699	0.720	0.742
RankPMemory	0.836	1.163	**0.700**	**0.725**	**0.745**
Methods	**Given15**				
	MAE	**RMSE**	**NDCG1**	**NDCG3**	**NDCG5**
PCC	0.780	0.999	0.710	0.732	0.752
EigenRank	0.876	1.285	0.722	0.741	0.758
RegPMemory	**0.774**	**0.987**	0.720	0.743	0.763
RankPMemory	0.803	1.066	**0.726**	**0.748**	**0.767**

datasets. Fig. 4.6 (a) and (b) show the NDCG evaluation for MovieLens (a) and Netflix (b) datasets. The configuration is Given15 for MovieLens and Given5 for Netflix.

From the two figures, it can be observed that as the iteration increases, all the measurements of MAE, RMSE and NDCG get better at first. This indicates that the optimization goal is effective. After some rounds, all of them get worse. This is because of the over-fitting problem in the training dataset. In MovieLens, the optimal point is around 35 iterations, and In Netflix, the optimal point is around 40 iterations. Therefore, we set the iteration number to these values in the following experiments.

Table 4.5: Performance of memory-based combination in Netflix

Methods	Given5				
	MAE	**RMSE**	**NDCG1**	**NDCG3**	**NDCG5**
PCC	0.760	0.936	0.753	0.767	0.781
EigenRank	1.033	1.687	**0.792**	0.775	0.793
RegPMemory	**0.753**	**0.918**	0.760	0.770	0.783
RankPMemory	1.005	1.600	0.790	**0.776**	**0.794**
Methods	**Given10**				
	MAE	**RMSE**	**NDCG1**	**NDCG3**	**NDCG5**
PCC	0.776	0.982	0.727	0.738	0.755
EigenRank	1.028	1.645	0.745	0.750	0.758
RegPMemory	**0.767**	**0.954**	0.730	0.746	0.759
RankPMemory	0.882	1.235	**0.755**	**0.755**	**0.767**
Methods	**Given15**				
	MAE	**RMSE**	**NDCG1**	**NDCG3**	**NDCG5**
PCC	0.838	1.161	0.707	0.710	0.712
EigenRank	1.046	1.685	0.744	0.732	0.740
RegPMemory	**0.819**	**1.079**	0.726	0.729	0.739
RankPMemory	1.020	1.605	**0.747**	**0.735**	**0.744**

Sensitivity Analysis

In this section, we make sensitivity analysis for the combination parameters α_1 and α_2. As discussed before, α_1 controls the weight of the regression objective function and α_2 controls the weight of the ranking objective function. Although the two functions are in the same scale and have the same optimal value 0, they indeed have different meanings. Thus in the combination, it is unnatural to have $\alpha_1 + \alpha_2 = 1$. Another problem is that for evaluation purpose, we have both regression metrics and ranking metrics, and adjusting the parameters will influence the metrics unequally. Thus in the combination, we train two models, regression-prior model (denoted as RegPModel) and ranking-prior model (denoted as RankPModel). In regression-prior model, we fix the weight $\alpha_1 = 1$ and adjust α_2 in the range $[0, +\infty]$. We will choose the value of α_2 to adapt to the best

regression performance. In the same way, in the ranking-prior model, we fix the weight of $\alpha_2 = 1$ and adjust the value of α_1 to adapt to the best ranking performance.

The sensitivity analysis of regression-prior model on both regression and ranking metrics for MovieLens and Netflix (Given15) is shown in Fig. 4.7(1)(2)(3)(4). $\alpha_1 = 1.0$ is fixed and the weight of α_2 is adjusted in the range $[0, 1]$ with an interval of 0.1. It can be observed that performance for all the metrics gets better at first and then gets worse. We omit the range $[1, +\infty]$ because the trend is similar. The better performance indicates that the combination is effective in improving the performance on both kinds of metrics. Furthermore, as show in figure, there is an optimal point that can achieve the best performance. Since it is a regression-prior model, we choose the value $\alpha_2 = 0.2$ to adapt to the regression performance in MovieLens and $\alpha_2 = 0.1$ in Netflix.

The sensitivity analysis of ranking-prior model on both regression and ranking metrics for MovieLens and Netflix (Given15) is shown in Fig. 4.7(5)(6)(7)(8). $\alpha_2 = 1.0$ is fixed and the weight of α_1 is adjusted in the range $[0, 1]$ with an interval of 0.1. It can be observed that the ranking performance gets better first and then gets worse. The regression performance gets better first and then gets stable. We also omit the rage $[1, +\infty]$ because the trend is similar. This figure can also indicate that the combination is effective in improving both regression and ranking performance. We choose $\alpha_1 = 0.1$ to adapt to the ranking performance in MovieLens and $\alpha_1 = 0.4$ in Netflix.

Overall Performance

In the verification of overall performance, we compare the combined methods RegPModel and RankPModel with the single-criterion methods PMF and LMF. Table 4.2 shows the overall performance on MovieLens and Table 4.3 shows the overall per-

formance on Netflix.

The first observation is that for regression metrics, the RegPModel consistently outperforms PMF and for ranking metrics, the RankPModel consistently outperforms LMF. This indicates that the newly-added objective function is effective in improving the performance of the original metrics. In MovieLens dataset, for regression-oriented metrics, RegPModel outperforms PMF by 2.9% with MAE and by 6.2% with RMSE at best. For ranking-oriented metrics, RankPModel outperforms LMF by 3.8% with NDCG at best. In Netflix dataset, for regression-oriented metrics, RegPModel outperforms PMF by 4.6% with MAE and by 2.6% with RMSE at best. For ranking-oriented metrics, RankPModel outperforms LMF by 3.9% with NDCG at best.

The second observation is that for regression-prior combination RegPModel, the performance of ranking metrics is consistently improved, and for ranking-prior combination RankPModel, the performance of regression metrics is significantly improved. In MovieLens, RegPModel outperforms PMF by 5.0% with NDCG at best and RankPModel outperforms LMF by 17.0% with MAE and 25.0% with RMSE at best. In Netflix, RegPModel outperforms PMF by 1.6% with NDCG at best and RankPModel outperforms LMF by 17.0% with MAE and by 24.4% with RMSE at best. In many cases, the performance of RegPModel in ranking metrics is approaching LMF and the performance of RankPModel in regression metrics is approaching PMF.

4.6.3 Performance in Memory-based Fusion

Sensitivity Analysis

In this section, we make sensitivity analysis for the combination parameter α. Since after conversion for EigenRank, the combination values have the same interpretation thus only one

parameter α is utilized to control the regression weight (where $1 - \alpha$ is utilized to control ranking weight). Again, we train two models, regression-prior model (denoted as RegPMemory) and ranking-prior model (denoted as RankPMemory) to adapt to different evaluation metrics respectively.

The sensitivity analysis on both regression and ranking for MovieLens and Netflix (Given15) is shown in Fig. 4.7(9)(10)(11)(12). α is adjusted in the range $[0, 1]$ with interval of 0.05. Similar to model-based methods, it is observed that performance for all the metrics gets better at first and then gets worse. The better performance indicates that the combination is effective in improving the performance on both kinds of metrics. We choose $\alpha = 0.5$ for RegPMemory and $\alpha = 0.8$ for RankPMemory in MovieLens and we choose $\alpha = 0.4$ for RegPMemory and $\alpha = 0.9$ for RankPMemory in Netflix.

Overall Performance

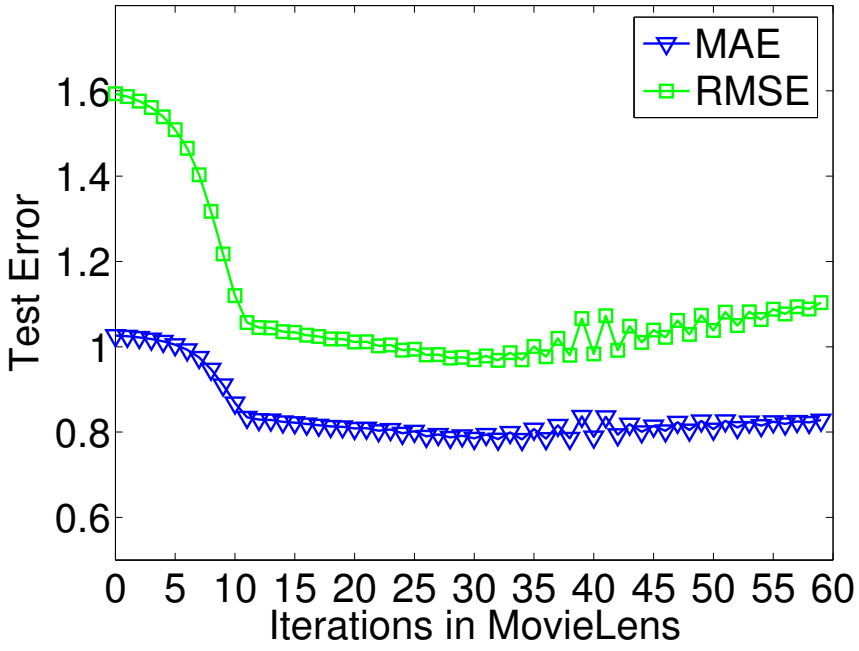
In the verification of overall performance, we compare the combined methods RegPMemory and RankPMemory with the single-criterion methods PCC and EigenRank. Table 4.4 shows the overall performance on MovieLens and Table 4.5 shows the overall performance on Netflix.

We have similar observations with model-based combination. First, for regression metrics, the RegPMemory consistently outperform PCC and for ranking metrics, the RankPMemory consistently outperform EigenRank. In MovieLens, for regression-oriented metrics, RegPMemory outperforms PCC by 6.7% with MAE and by 12.5% with RMSE at best. For ranking-oriented metrics, RankPMemory outperforms EigenRank by 1.3% with NDCG at best. In Netflix dataset, for regression-oriented metrics, RegPMemory outperforms PCC by 2.3% with MAE and by 7.0% with RMSE at best. For ranking-oriented metrics, RankPMemory outperforms EigenRank by 1.4% with NDCG at best.

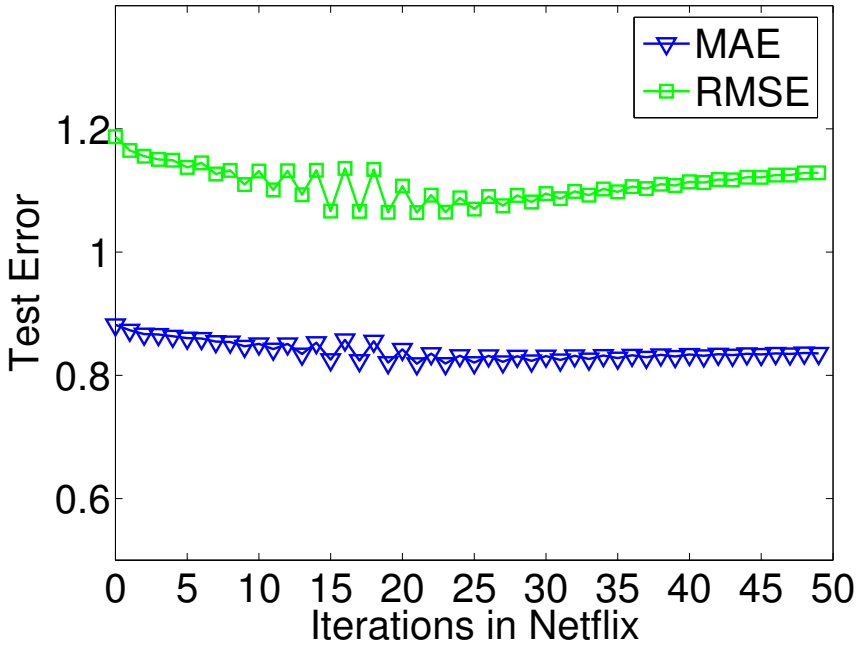
Secondly for RegPMemory, the performance of ranking metrics is consistently improved; and for RankPMemory, the performance of regression metrics is significantly improved. In MovieLens, RegPMemory outperforms PCC by 1.5% with NDCG at best and RankPMemory outperforms EigenRank by 8.3% with MAE and 17% with RMSE at best. In Netflix, RegPMemory outperforms PCC by 3.0% with NDCG at best and RankPMemory outperforms EigenRank by 14.2% with MAE and by 24.9% with RMSE at best. In many cases, the performance of RegPMemory in ranking metrics is approaching EigenRank and the performance of RankPMemory in regression metrics is approaching PCC.

4.7 Summary

In this work, we investigate the fusion of regression and ranking for multi-measure adaption in recommender systems. We choose competitive regression-oriented and ranking-oriented methods from both model-based algorithms and memory-based algorithms and propose combination methods. Experimental verification on two real-world datasets indicates that the combinations are effective in improving performance on both regression and ranking evaluation metrics.

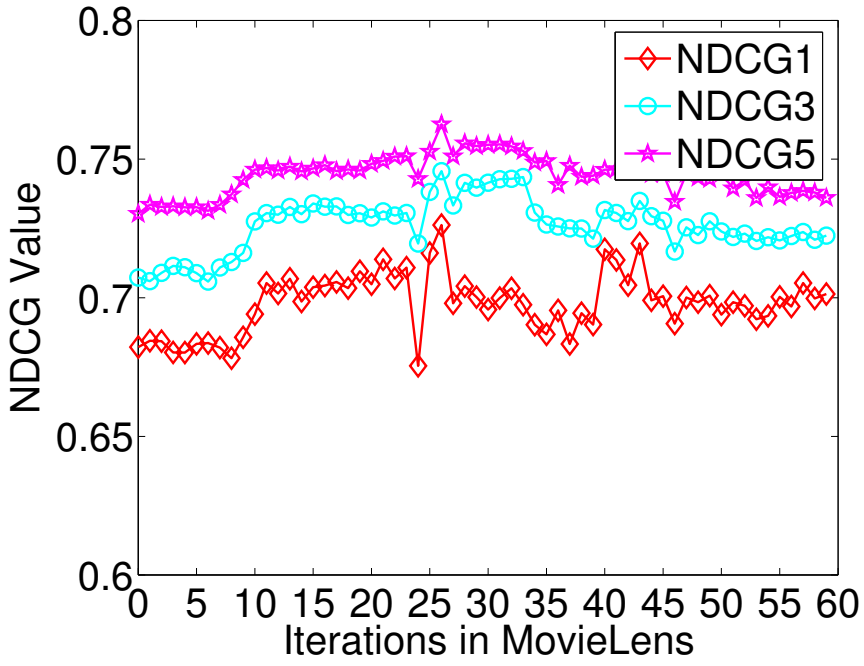


(a)

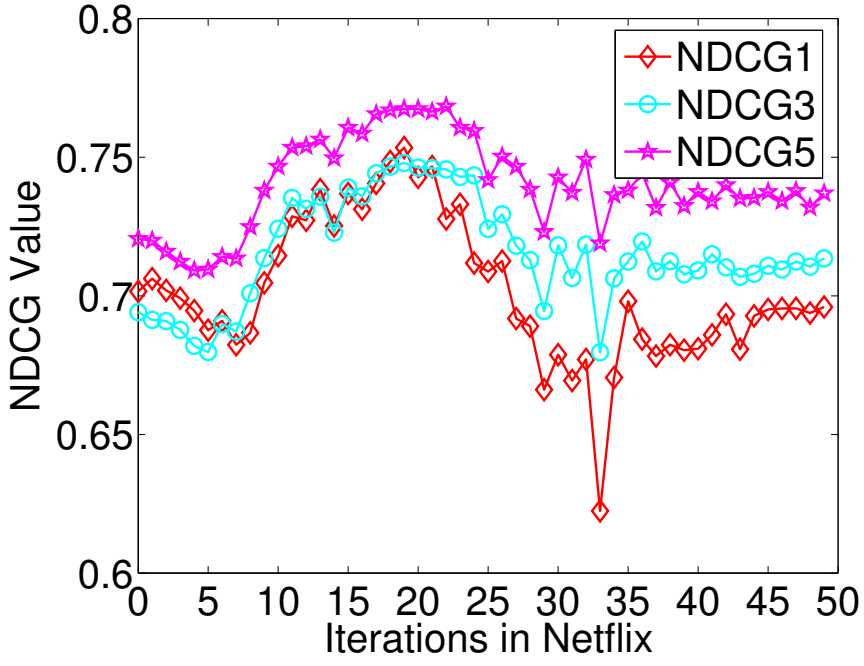


(b)

Figure 4.5: Convergence in model-based combination (test error)



(a)



(b)

Figure 4.6: Convergence in model-based combination (NDCG value)

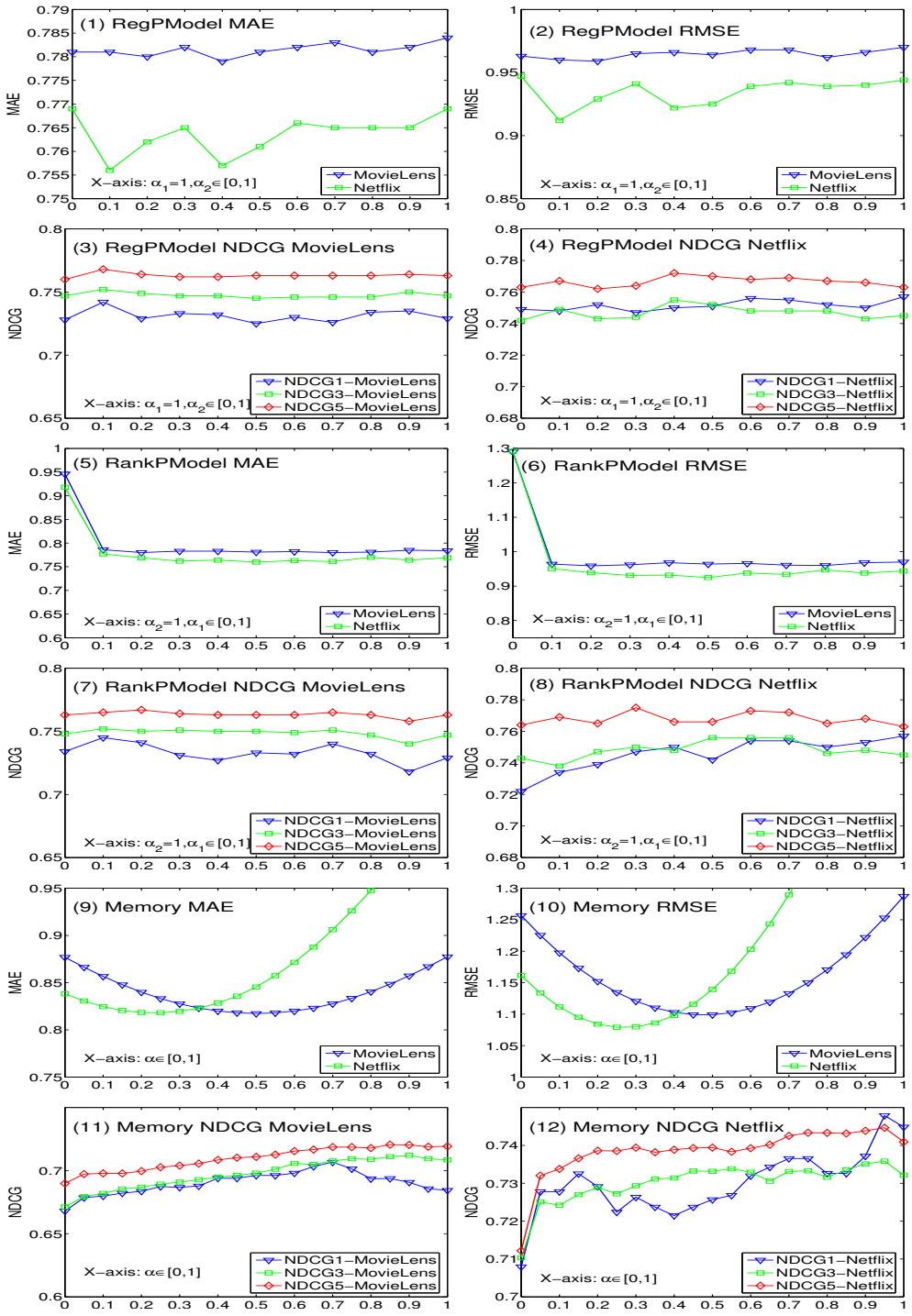


Figure 4.7: Sensitivity analysis of all combination methods

Chapter 5

Effective Fusion of Quality and Relevance for Multi-dimensional Adaption

5.1 Limitation of Single-dimensional Adaption Identification from Qualitative Analysis

Multi-dimension recommendation has been predicted as an important direction in the next generation of recommender systems [1, 20]. The concept "multi-dimension" here refers to some other recommendation dimensions besides quality evaluated from ratings. Typical dimensions include relevance, coverage, diversity, etc. Although one may argue that the evaluation of relevance, coverage, and diversity might also be contained in ratings, from previous work [36, 65], ratings reflect mainly for quality rather than other dimensions. A recommender system's success in multi-dimension recommendation should consider all the dimensions besides quality.

Successful recommender systems should simultaneously consider multi-dimensional performance. Previous work, however, is mainly focusing on a single dimension. The limitation is that the single-dimensional recommendation results may not adapt

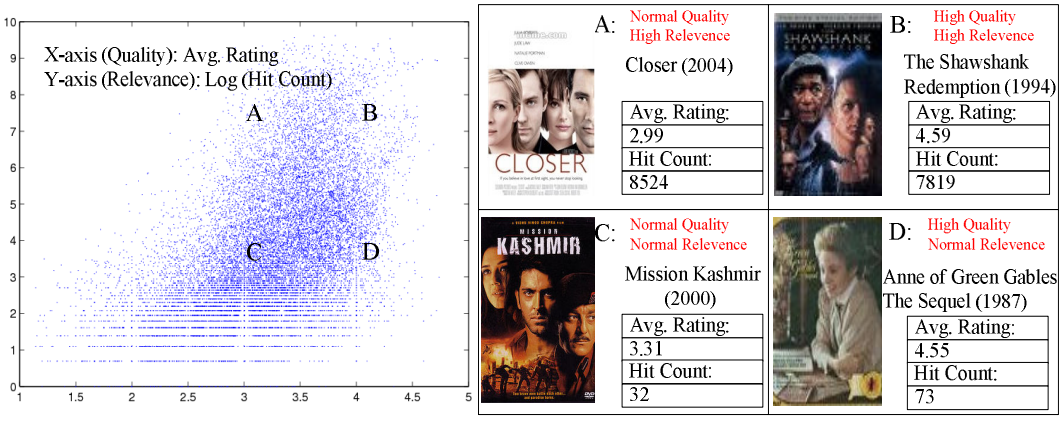


Figure 5.1: Distribution of items in relevance and quality

to other dimensions in many cases.

Intuitively some conflicts might happen from different dimensions, thus in multi-dimension recommendation it is important to investigate how each dimension impacts each other, how to combine multiple dimensions 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.

In this work, we study the interplay relationship of different dimensions and investigate how to combine previous single dimension approaches in multi-dimension recommendation. Specifically, we investigate the dimensions of quality and relevance as a preliminary work because they are more practical and both have enough previous work. In this section, we give a qualitative analysis to show that single-dimensional algorithms cannot adapt other dimensions.

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. 5.1 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) [17, 130], Aspect Model (AM) [67], PMF [128] and EigenRank [91] as quality-based algorithms; and within relevance-based algorithms, an association-based method [36] and a hitting-frequency-based method [149] 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. 5.2 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

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

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 dimensions. 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 advertising or selling business [131], 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 [42]. 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 [43]. 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-dimension recommendation is important.

5.2 Integrated Metric of Quality and Relevance for Multi-dimensional Adaption

For considering both quality and relevance for multi-dimension recommendation, we linearly combine quality-based and relevance-

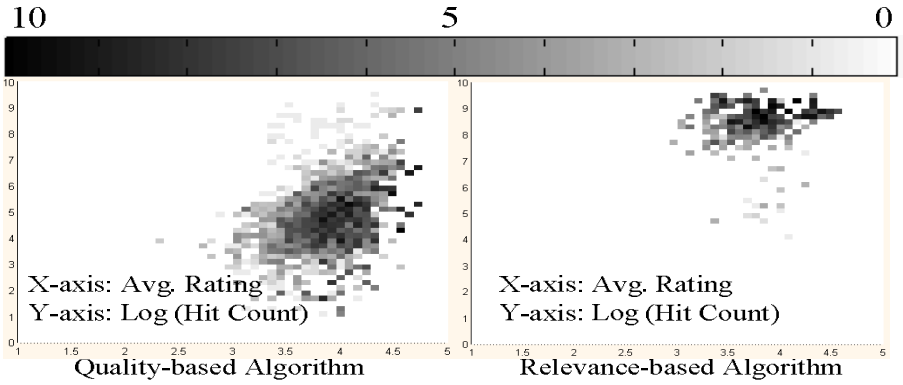


Figure 5.2: Distribution of recommended results

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. Relevance-based metrics, on the other hand, measure the likelihood that an item will be hit. 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 [91] and relevance-based NDCG [56]. In fact, there are many ways for the integrated metric; however, the comparisons of metrics are beyond the scope of this work. 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 [56, 91]. 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 [91])

$$NDCG_{P-quality} = \frac{1}{U}\sum_{u}^{U} Z_u \sum_{p=1}^{P} \frac{2^{r_{u,p}} - 1}{\log(1+p)}, \tag{5.1}$$

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 [91], uncertain ones in the rank are removed before calculation. Relevance-based NDCG at position P, on the other hand, is defined as (refering to [56])

$$NDCG_{P-relevance} = \frac{1}{U}\sum_{u}^{U} Z_u \sum_{p=1}^{P} \frac{2^{h_{u,p}} - 1}{\log(1+p)}, \tag{5.2}$$

where $h_{u,p}$ is a binary value function indicating whether user u has hit the item at position p. In this work, we follow the same idea in [36]: 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

$$NDCG_{I-linear} = \lambda * NDCG_Q + (1-\lambda) * NDCG_R. \tag{5.3}$$

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.

5.3 Fusion Approaches for Multi-dimensional Adaption

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 [47]. The second method is RankComb, a rank-based integration, which is to combine the rank results of different recommendation algorithms linearly by Borda count [40]. 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 Continuous-time Markov Process (CMAP), to solve above problems. By further employing queueing system theory, CMAP has a intuitive interpretation without losing quantity information.

5.3.1 Rationale of Basic Components 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 regression-oriented (e.g., PMF [128]) and ranking-oriented (e.g., EigenRank [91]). Regression-oriented methods predict ratings. Ranking-oriented methods, on the other hand, predict the rank of recommended items based on directly modeling the preference order of arbitrary two items [91]. Since the integrated metric is based on ranking, thus we choose EigenRank as the fundamental method in this work. This model

is proven to outperform many classical methods [91]; and we have also demonstrated that it outperform PMF in quanlity-based NDCG through experiments. EigenRank model is based on random walk theory, which is a special case of Discrete-time Markov Process (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

$$P\{X_{t+1} = i_{t+1} | X_0 = i_0, X_1 = i_1, ..., X_t = i_t\} = P\{X_{t+1} = i_{t+1} | X_t = i_t\}. \quad (5.4)$$

The stationary distribution of this DMP is defined as

$$\pi = \pi * P, \quad (5.5)$$

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

$$p_{ij} = p(j|i) = \frac{e^{\psi(j,i)}}{\sum_{j \in S} e^{\psi(j,i)}}, \quad (5.6)$$

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

$$\psi(i, j) = \frac{\sum_{v \in N_u^{i,j}} s_{u,v} \cdot (r_{v,i} - r_{v,j})}{\sum_{v \in N_u^{i,j}} s_{u,v}}. \quad (5.7)$$

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) [100] similarity.

Within relevance-based algorithms, we choose an association-based method [36] and a hitting-frequency-based method [149]. Association and hitting frequency have been demonstrated as

two competitive features in relevance-based approaches. Therefore, we combine association and hitting frequency into the EigenRank model to form an unified recommendation approach. 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 [36]. 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 [80, 149].

5.3.2 Fundamental Fusion Approach Based on Linear Combination

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

$$S_{LinearComb} = w_1 F(EigenRank) + w_2 F(Assoc) + w_3 F(Hit - freq). \quad (5.8)$$

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

$$F(x) = 1/(1 + \exp(-x)). \quad (5.9)$$

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 sta-

tionary 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 [47].

5.3.3 Fundamental Fusion Approach Based on Ranking Combination

To solve the problem of LinearComb, we propose another fundamental combination method RankComb, which combines the ranks from different methods by employing Borda count [40]. 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

$$BC_{item} = 1/position(item). \tag{5.10}$$

In Borda count, if the item ranked higher, the value would be larger. Also, the higher position is more important than the lower position. Then, the item's BC values from different algorithms can be linearly combined as a new recommendation score as

$$\begin{aligned} BC_{RankComb} = w_1 BC_{EigenRank} + w_2 * BC_{Assoc} \\ + w_3 * BC_{Hit-freq}. \end{aligned} \tag{5.11}$$

The final results will refer to this new BC value. The weights are adjusted manually. 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.

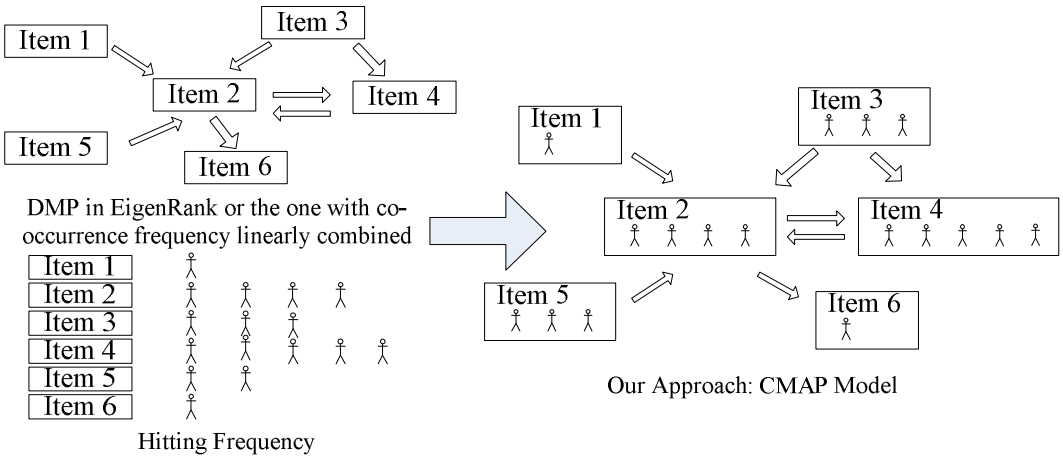


Figure 5.3: An overview of CMAP approach

5.3.4 Effective Fusion Approach Based on Continuous-time Markov Process (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 approach 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.

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

$$s'_{u,v} = \frac{\overrightarrow{R}_u * \overrightarrow{R}_v}{\|\overrightarrow{R}_u\| * \|\overrightarrow{R}_v\|}, \tag{5.12}$$

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 [36], we define a co-occurrence function of two items i and j as

$$\xi(i,j) = \frac{Freq(ij)}{Freq(i) * Freq(j)^{\beta}}, \tag{5.13}$$

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 [17, 36]. Thus the term $Freq(j)^{\beta}$ is added following the idea of inverse document frequency [112] to adjust the weight. Therefore, the transition matrix P' can be defined as

$$p'_{ij} = p'(j|i) = \frac{\xi(i,j)}{\sum_{j \in S} \xi(i,j)}. \tag{5.14}$$

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

$$P_{new} = P * \alpha + P' * (1 - \alpha). \tag{5.15}$$

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. 5.3. 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 $0 \leq t_0 < t_1 < ... < t_n < t_{n+1}, i_k \in S, 0 \leq k \leq n+1$,

$$P\{X_{t_{n+1}} = i_{n+1} | X_{t_0} = i_0, X_{t_1} = i_1, ..., X_{t_n} = i_n\} = P\{X_{t_{n+1}} = i_{n+1} | X_{t_n} = i_n\}. \quad (5.16)$$

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

$$P\{X_{s+t} = j | X_s = i\} = P\{X_t = j | X_0 = i\} = p_{ij}(t). \quad (5.17)$$

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

$$\begin{cases} q_{ij} = p'_{ij}(0) = \lim_{t \to 0} \frac{p_{ij}(t)}{t}, i \neq j; \\ q_{ii} = -\lim_{t \to 0} \frac{1 - p_{ii}(t)}{t}. \end{cases} \quad (5.18)$$

Under such description, it can be proven [148] that the staying time at each state follows an exponential distribution

$$P(\tau > t | X_{pre} = i) = \exp(q_{ii} t), \quad (5.19)$$

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

$$\tau = inf\{t : t > 0, X_t \neq X_{pre}\}. \quad (5.20)$$

It can also be proven [148] that

$$P[X_\tau = j | X_0 = i] = \frac{q_{ij}}{-q_{ii}}. \quad (5.21)$$

From Eq. (5.19) and Eq. (5.21), 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.

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. 5.3, 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 [6] that

$$P(T_g \leq x) = 1 - \frac{v}{u} \exp^{-(u-v)x} . \tag{5.22}$$

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

$$P(T_g \leq x | \textit{the queue exists}) = 1 - \exp^{-(u-v)x} . \tag{5.23}$$

Thus we propose to model hitting frequency using

$$q_{ii} = -(u - v_i). \tag{5.24}$$

If u is larger, the variance of waiting time conditions becomes smaller, which means the staying time has a weaker impact on

Algorithm 5 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. (5.24)
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. (5.15)
6: Calculate stationary distribution of CMAP according to Eq. (5.25)
7: Rank the items based on the probabilistic score of a stationary distribution of CMAP
8: Remove the rated items
9: **end for**

the final stationary distribution. If u is small, then the opposite is true.

Algorithms

We still employ stationary distribution to decide the preference score of an item. According to [91, 148], the stationary distribution π of CMAP can be solved using the following equations:

$$\begin{cases} \pi_i = \frac{\frac{\widetilde{\pi}_i}{-q_{ii}}}{\sum_{j=1}^{S} \frac{\widetilde{\pi}_j}{-q_{jj}}}; \\ \\ \widetilde{\pi}_j = \sum_{i \in S} \widetilde{\pi}_i \frac{q_{ij}}{-q_{ii}}. \end{cases} \tag{5.25}$$

The details of the algorithm are shown in Algorithm 5.

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. (5.25) 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.

5.4 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 dimensions in recommendations. The second part is to evaluate the recommendation performance of our proposed approach. The third part is to do the sensitivity analysis of CMAP.

5.4.1 Datasets

In this work, 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

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

Table 5.1: 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%

an integer value on the scale of 1 to 5, with higher value indicating better satisfaction. More statistics are shown in Table 5.1. In MovieLens, referring to the experimental setup in [96, 163], 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.

5.4.2 Limitation of Single-dimensional Adaption Verification from Quantitative Analysis

In this section, quantitative analysis of competitive quality-based and relevance-based algorithms on multiple dimensions is conducted. The purpose is to evaluate quality-based algorithms' performances on relevance-based metric and relevance-based al-

Table 5.2: Performance on quality-based NDCG

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

gorithms' performances on quality-based metric. In the experiments, two quality-based algorithms and two relevance-based algorithms are chosen. Quality-based algorithms include PMF [128] and EigenRank [91]. Relevance-based algorithms include the association-based method (Assoc) [36] and a hitting-frequency-based method (Freq) [149]. The experiments are conducted on both MovieLens and Netflix. We only report the results for MovieLens in Table 5.2 and Table 5.3 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 outper-

Table 5.3: Performance on relevance-based NDCG

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

form 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.

5.4.3 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) Qual-

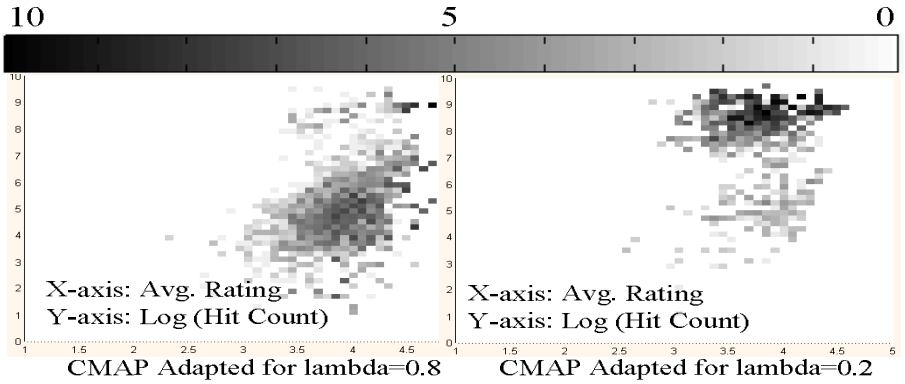


Figure 5.4: Distribution of recommended results of CMAP

itatively, whether the incompleteness problem in each single-dimension approach can be solved by our approach? 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 approach 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 [91] 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 [36] and the hitting-frequency-based method [149] have their own advantages in different cases as shown before, we choose the best result from them as our baseline method.

Fig. 5.5(a) shows the overall performance on MovieLens; and Fig. 5.5(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 5.4 and Table 5.5.). In both these two figures, we can observe that the three combination methods outperform the two single-dimension 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. 5.4 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 approach is effective in solving the incompleteness limitation of single-dimension 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

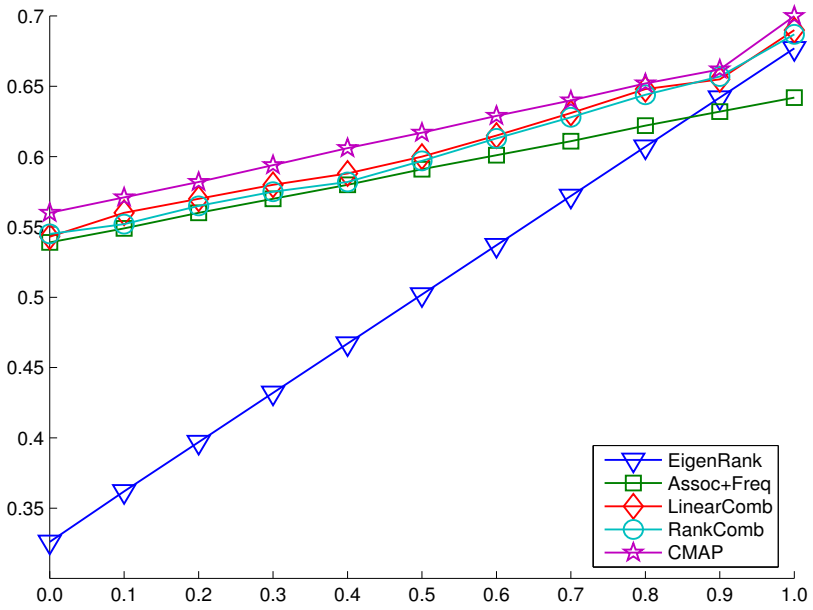
quality and relevance are emphasizing different aspects, there is some correlative relation between them. In these two datasets, if an item is relevant to a user, it will have great chance to have good quality; but the opposite is not true that many high-quality items do not attract that many users. 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.

5.4.4 Sensitivity Analysis

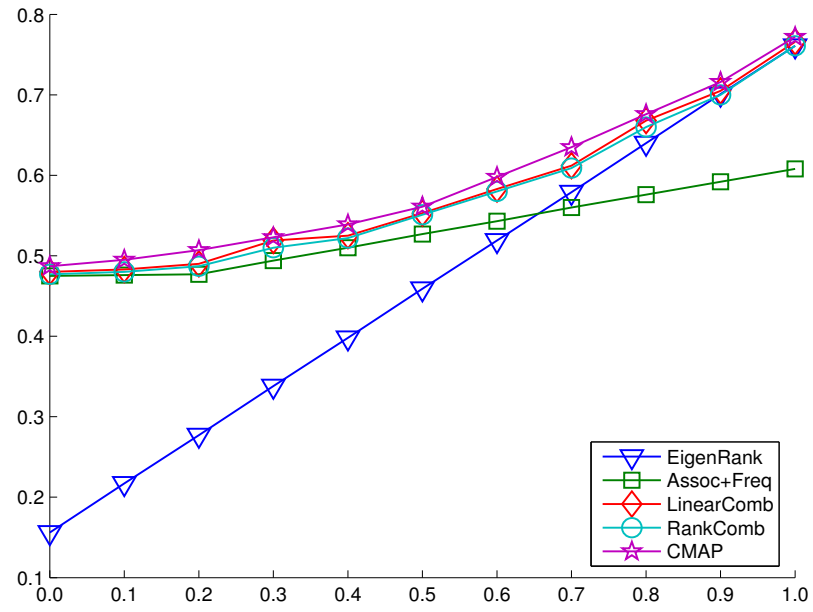
There are two important parameters in our approach: α in Eq. (5.15) and u in Eq. (5.24). α balances CMAP's probability transition matrix between rating preference order and association feature. It scales from 0 to 1. When $\alpha = 1$, the transition matrix is built from quality-based information only; 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. 5.6(a) shows the impact of α on MovieLens, given 10 ratings as history, with $\lambda = 0.8$ and $u = 20$ and Fig. 5.7(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. 5.6(b) shows the impact of u on MovieLens, given 10 ratings as history, with $\lambda = 0.8$ and $\alpha = 0.6$. Fig. 5.7(b) shows the impact of u on Netflix, given 10 ratings as history, with $\lambda = 0.2$ and $\alpha = 0.4$.

5.5 Summary

In this work, we make a preliminary fusion work for multi-dimensional adaption in recommender systems. We take quality and relevance as two dimensions in recommender system for analysis. We study the interplay relationship of their impact to each other and show that both quality-based and relevance-based methods cannot perform well in the other dimensions. As the first ever solutions, we propose an integrated metric considering both dimensions. Then we investigate how to combine previous work to adjust the new metric under the concept of multi-dimensional recommendation. We propose a CMAP approach 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 approach has linear computational complexity. Thus from a technical standpoint, we believe the work will be helpful in improving recommender system in real applications.

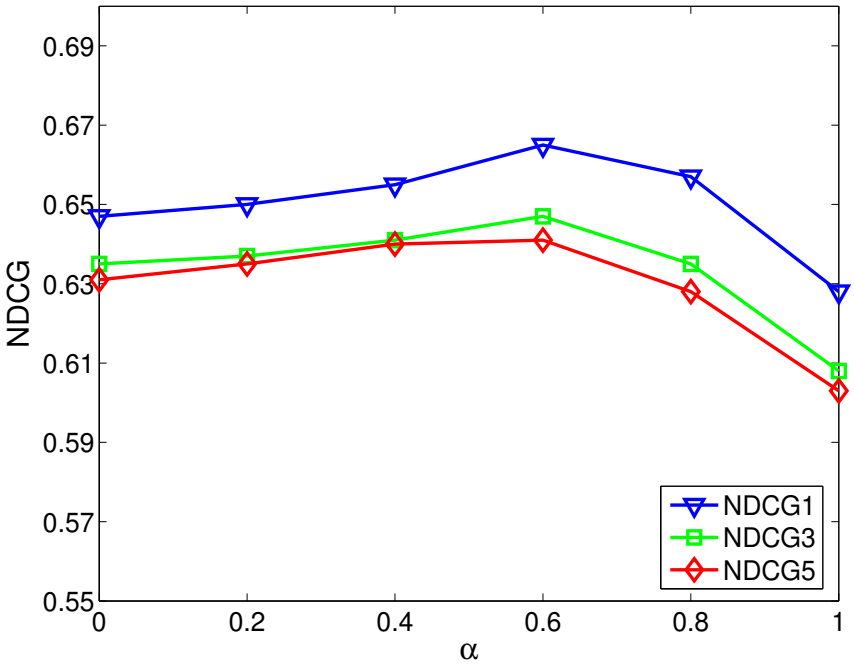


(a) MovieLens

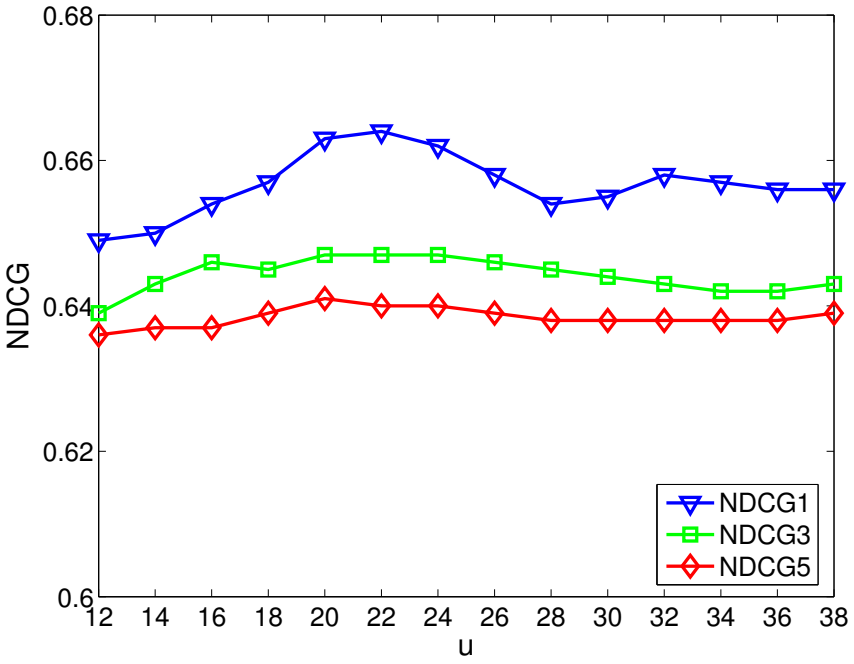


(b) Netflix

Figure 5.5: Recommendation performance

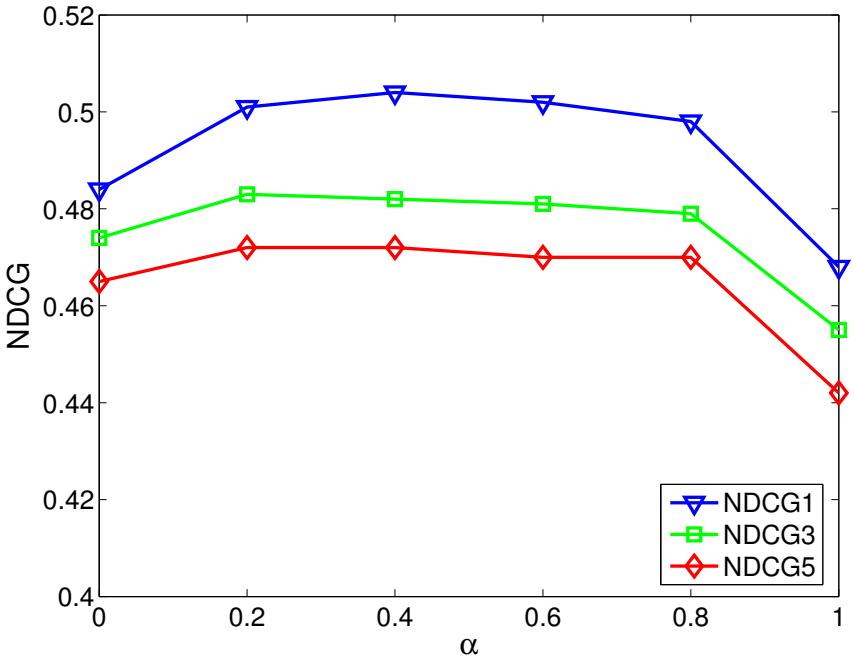


(a) Impact of α

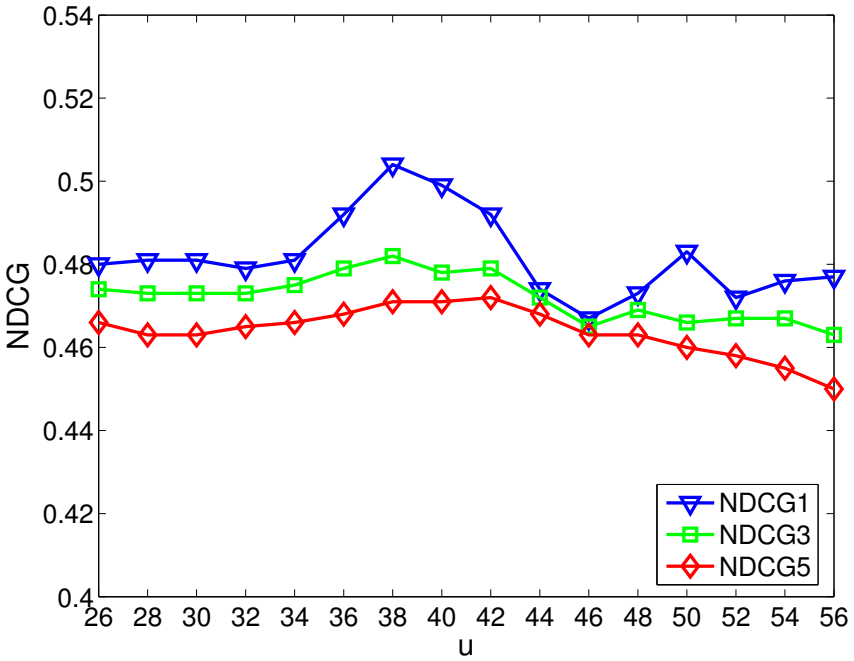


(b) Impact of u

Figure 5.6: Impact of parameters of CMAP in MovieLens



(a) Impact of α



(b) Impact of u

Figure 5.7: Impact of parameters of CMAP in Netflix

Table 5.4: Overall performance for other settings in 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 5.5: Overall performance for other settings in 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**

Chapter 6

Impression Efficiency Optimization for Recommender Systems

6.1 Commercial Intrusion Problem from Low Impression Efficiency

Recommendation impression efficiency is another important issue, along with the development of algorithms in recommender systems. Recommendation impression efficiency means how much revenue E-business companies can obtain by impressing a recommendation result to users. The reason why optimizing the impression efficiency is important is that over-quantity recommendation would have commercial intrusion to users. Thus optimizing impression efficiency means to optimize the profit on the constraint that the total impression number is limited in a certain range.

If too many recommendation results are pushed to users, users' satisfaction would be destroyed. Currently, most recommender systems are supported by E-business companies. While these companies provide Web services to users, they also expect users would have more commercial behavior for their business. Thus the recommendation service in such an environment can

be seen as a commercial behavior from E-business companies to users.

There are many evidences to support the existence of commercial intrusion in recommender systems. Take the example of sponsored advertisements (ads) recommendation in sponsored search, we list the following three supporting materials: 1) Users have reported to show bias against sponsored search results after they know its commercial insight [99]. 2) From the user study in [71], when sponsored results are as relevant as the organic results, more than 82% of users will see organic results first. 3) Organic results have also demonstrated to gain much higher click through rate (CTR) than sponsored search results [33]. Thus if the ads are irrelevant to users' search intent, to show less ads or even not to show any ads is better than to show a full rank of ads [18]. From the research in [21], irrelevant ads will have the effect to "train" the users to ignore ads in the result page. Thus if the impression efficiency is not carefully considered, in a long-term, users would not trust the recommendation service and finally it will decrease the utility of advertisers and search engines.

In this chapter, we utilize sponsored search as a specific recommendation application to investigate the impression efficiency optimization problem for recommender systems.

6.2 Background of Advertisements Recommendation in Sponsored Search

Sponsored search has attracted more and more attention for both research and industry community since web advertising has become a large business industry nowadays. According to a report from eMarketer[1], advertisers in US have spent $21.4

[1] www.emarketer.com, October 2007

billion on web advertising in 2007, and this number is predicted to be increased to $42 billion in 2011. In web advertising, sponsored search is a major component, making up around 40% (also from eMarketer) of the total revenue. Sponsored search is the main revenue source for search engines. Usually, the pay-per-click mechanism is commonly used that advertisers would pay an amount of money to the search engine company once their ads shown in sponsored search results are clicked by users.

The majority of research in sponsored search is focusing on matching relevant ads for queries. These work can be divided into two streams, learning a function to directly predict an ad rank list given a query [26, 29, 176] and learning to predict the click-through rate (CTR) for query-ad pairs as an intermediate step for matching relevant ads [11]. In the former stream, some work focus on improving traditional models by exploring new features such as query expanding [19, 120, 155], user behavior [8], personalization [164], etc. In the latter stream, the work can be divided into estimating CTR for frequent queries based on click model [162, 172] and predicting CTR for rare queries based on regression and classification model [35, 55, 125]. Other work includes [15, 52, 167].

Impression efficiency optimization is also an important research issue in sponsored search, but currently, most relevant work is focusing on its foundation task, commercial revenue estimation of a query-ad pair. A query-ad pair refers to a query and its displayed ad. The revenue value means how much revenue is expected to obtain by showing such a query-ad pair. Precise estimation of revenue is a preliminary task for further strategy to optimize the impression efficiency. Typical revenue estimation methods include: (1) CTR-based estimation [58]; (2) relevance-based estimation [18]; and (3) advertisability-based estimation [114]. In CTR-based estimation [58] method, click-models are employed to calculate the CTR of the query-ad pair,

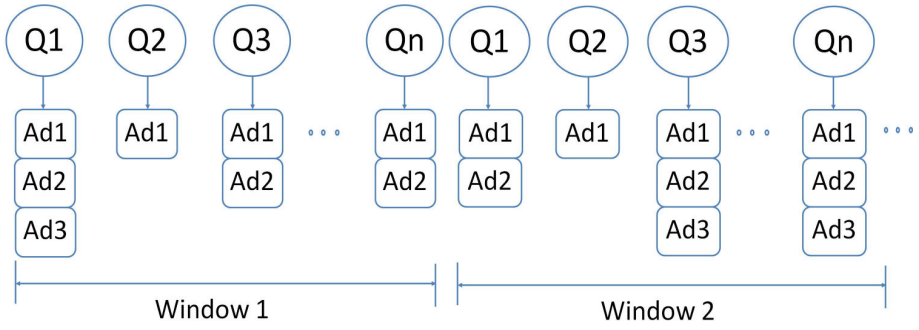


Figure 6.1: Problem illustration for the impression efficiency optimization

and then the expected revenue can be obtained by multiplying the bid price of the ad. In relevance-based estimation [18], a classifier based on Support Vector Machines (SVM) is trained to identify the relevance score of the query and the ad as the revenue estimation. In advertisability-based estimation [114], an advertisability score is calculated from the statistics of click through log data as the estimation of revenue. More details of the three kinds of methods can be found in the original papers.

Although the expected revenue of ads can be estimated by previous work, the research issue of the strategy to optimize the ads impression efficiency has not been formulated and thoughtfully explored. In this work, as the first ever solution, we formulate the problem in sponsored search in the framework of secretary problem. Then we explore approaches to solve it.

6.3 Problem Formulation for Impression Efficiency Optimization

6.3.1 Preliminary Knowledge

For each query q_i, the search engine generate a ranked list of relevant ads. The ranking is decided by a bidding process and

a combined consideration of many factors. The mechanism for generating the rank is out scale of the work and we assume the rank is fixed. Usually, there is a maximum number of ads to show. In this work, we only consider north ads without loss of generality. The maximum of ads that can be shown is 3. The ads after the third position in the rank will be ignored.

6.3.2 Problem Formulation

Since the goal of the impression efficiency optimization is to reduce commercial intrusion to users, we first quantify the commercial intrusion. Intuitively, if more ads are shown for each query, users would suffer from more commercial intrusion. Thus in this work, we propose to employ the rate λ, the average ad impression number per query, as the quantitative description for commercial intrusion to users as shown in Def. 1. Although in rare cases, users may seem to take interests in some of the ads and click them, from the work in [71, 99], the commercial intrusion in this case still objectively exists and users would still prefer to show organic results only.

Definition 1. *The* ***intrusion rate*** λ *for sponsored ads recommendation is defined as the average ad impression number per query.*

The impression efficiency and its optimization problem is defined as follows. As shown in Fig. 6.1, for a window of N queries, they arrive at the search engine in a sequence. When a query arrives, a fixed function $f(q_i, ad_j)$ can be utilized to calculate the commercial revenue estimation of the ad at the j^{th} position. If there are less than three ads, the missing position has the estimated revenue value of zero. Before the next query arrives, an impression function $I(q_i)$ has to decide how many ads should be shown for current query. The impression efficiency is defined in the following way.

Definition 2. *The **impression efficiency** at λ for sponsored ads recommendation is defined as*

$$\frac{\sum_{i=1}^{N}\sum_{j=1}^{I(q_i)} f(q_i, ad_j)}{\sum_{i=1}^{N} I(q_i)}, \tag{6.1}$$
$$Sub.\ to\ \sum_{i=1}^{N} I(q_i) \leq N * \lambda.$$

Therefore, the problem is to find the impression function that can maximize the sum of estimation values of shown ads in the window under the constraint that the total number of ads is less than $N * \lambda$.

Definition 3. *The problem of **impression efficiency optimization** for sponsored ads recommendation is defined as*

$$\max_{Imp(q)} \sum_{i=1}^{N}\sum_{j=1}^{I(q_i)} f(q_i, ad_j), \tag{6.2}$$
$$Sub.\ to\ \sum_{i=1}^{N} I(q_i) \leq N * \lambda.$$

From this problem definition, we can see the difference between our work and previous work. Previous work is mainly focus on exploring function f to estimate the expected revenue of the query-ad pairs. But in our work, we focus on exploring function I to optimize the total revenue, which is a research issue that has rarely been investigated in sponsored ads recommendation.

In this work, we make the assumption that within a query window of N, queries arrive randomly. Thus the problem defined in this work is under the secretary problem framework. However, it is different from any traditional secretary problem. The most basic secretary problem is to select the best secretary from

a sequence of N applicants. After one arrives, its quality can be known and a decision should be made immediately without repentance. The optimal policy for this problem can achieve the probability of $1/e$ for success. A similar secretary problem with our problem is the multiple-choice secretary problem. That is to select k secretaries from the applicants and optimize the sum of their quality score. The condition in our problem is more complex. When a query arrives, three ads arrive as a group, rather than one. Furthermore, in the decision, there are further constraints. We can only select the first s ads ($s \in [0, 1, 2, 3]$). Since the ranking is decided by a bidding process from advertisers and it is fixed, we cannot show the third ad without showing the second. Thus this problem has never been investigated under the secretary framework. We call the problem constrained 3-tuple multi-choice secretary problem. More details of secretary problems can be found in [9, 31, 45, 49, 48, 41]

6.3.3 Evaluation Metric

The main challenge of this secretary problem is that we do not know all the estimated revenues in the window when making decisions. If we know them, then the optimal decision can be solved by simple linear programming. We suppose the "best solution" is the decision made in this way. As the first ever solution for measurement, we employ the error distance rate of the algorithm solution to the best solution as the evaluation metric. Suppose for an algorithm I to this secretary problem, the sum of all estimated revenue obtained is $\sum_{i=1}^{N}\sum_{j=1}^{I(q_i)} f(q_i, ad_j)$, and suppose the best solution is I_{best}, the error distance rate is defined as

$$error\ distance\ rate = \frac{\sum_{i=1}^{N}\sum_{j=1}^{I_{best}(q_i)} f(q_i, ad_j) - \sum_{i=1}^{N}\sum_{j=1}^{I(q_i)} f(q_i, ad_j)}{\sum_{i=1}^{N}\sum_{j=1}^{I_{best}(q_i)} f(q_i, ad_j)}. \tag{6.3}$$

Table 6.1: Statistics of the queries

Query Freq.	**# Unique Query**	**# Session**	**Avg. CTR**
1	1695146	1695146	0.0212
2	1058697	1342854	0.0153
3-4	772810	1275067	0.0141
5-8	262555	925065	0.0151
9-17	128304	880444	0.0162
18-32	47981	685582	0.0179
33-221480	48582	4427998	0.0201

Table 6.2: Statistics of the ads

Ads Freq.	**# Unique Ad**	**# Impression**	**Avg. CTR**
1	26267	26267	0.0257
2	14689	29378	0.0222
3-4	17539	60146	0.0216
5-8	18058	113186	0.0208
9-17	18092	223102	0.0191
18-32	12786	306201	0.0190
33-82942	36365	17252570	0.0178

The error rate ranges from 0.0 to 1.0. Please notice here the metric is a measurement of error, so the smaller the value is, the better the performance is.

In the metrics, the revenue scores are the estimated values done by previous work, because in this work, we focus on exploring the strategy for impression efficiency as an independent problem rather than the revenue estimation. In the experiments, we also show the error distance rate of real revenue obtained between I and I_{best}. But it is not utilized as a metric. We show this only as a demonstration that the strategies of impressing ads are practical in real applications.

6.4 Dataset and Experimental Setup

6.4.1 Dataset

Our dataset is from a search engine company Sogou[2] from the mainland China. Sogou is the second largest search engine company. The dataset is the click-through log collected in part of its sponsored search module. The queries are focused at educational topic, such as "College English Test Four", etc. The data is collected from October 1st to December 1st and it contains 18,010,850 sessions in total. It has 6 field information, including the query, ad title, ad body, ad bid terms, ad bid price, and whether it is clicked. In this example, three ads are generated from the search engine. Although the time-stamps are not included, the data is collected by the time sequence. Statistics of the results are summarized in Table 6.1 and Table 6.2. The average number of clicks per query is quite low for both queries and ads in general, with 0.02515 at position 1, 0.009369 at position 2 and 0.006844 at position 3.

6.4.2 Experimental Setup

In this experiments, we set the window length N=500,000 without loss of generality. For a certain λ, the maximum number of ads to show is $\lambda * N$. Experiments are conducted on different configurations of λ. Its range is from 0.5 to 1.15 with the interval of 0.05. The average error distance rate is calculated among all the query windows as the result of an algorithm.

For ad revenue estimation function f in this work, we employ CCM [58] model for CTR prediction. The revenue estimation of a query-ad pair is the CTR of the ad multiplied by its bid price. The advantage of this method is that the position-bias is considered, which makes the estimation more accurate.

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

6.5 A Preliminary Assumption for All Methods

To make the problem simple, we assume, for a query, the estimation revenue of the ad at the latter position is always smaller than the revenue of the ad at the former position. There are two supportive points for this assumption: 1) The position-bias problem exists in sponsored search. People are likely to click the ads at the former position more than the ads at the latter position. 2) In ranking the ads, to place the ad with more estimated revenue at the former position is also the goal of the ranking algorithm in search engine. The statistics of this dataset can also verify this statement.

6.6 Unstable Problem in Static Method for Impression Efficiency Optimization

6.6.1 Static Method Description

An intuitive way to solve the problem is to employ static methods. A fixed threshold value can be learned from history data to determine whether the current query-ad pair has the estimated value large enough to show. The assumption of this kind of methods is that the distribution of the estimated revenue at each window is stable over times.

A direct static method is shown in Algorithm 6. In this algorithm, N is the window size, X_i is a three dimension vector (X_{i1}, X_{i2}, X_{i3}) with X_{ij} denoting the estimated revenue of the ad for query i at position j. An ad can be shown if all the estimated revenue values of its previous ads (including itself) are larger than the threshold. In learning the threshold, we can use the data in the previous window by selecting the k^{th} largest value in the window, as shown in Algorithm 7. The complex-

Algorithm 6 Static Method

Input: A sequence of N queries together with estimated revenue for their ads $X = X_1, X_2, ...X_N$
Threshold: to determine whether to impress an ad
k : the maximum number of ads to select
Algorithm:

```
Variable remainToSelect = k
for i = 0 to N-1 do
  if X_i1 > Threshold then
    Impress ad_i1 to the user
    remainToSelect=remainToSelect-1;
    if X_i2 > Threshold then
      Impress ad_i2 to the user
      remainToSelect=remainToSelect-1;
      if X_i3 >Threshold then
        Impress ad_i3 to the user
        remainToSelect=remainToSelect-1;
      end if
    end if
  end if
end for
```

Output: The strategy for impressing ads for current N queries.

ity of the learning is mainly from the sorting of all the revenue values, thus it is $O(n \log n)$. The complexity in application is linear complexity $O(n)$.

6.6.2 Experimental Verification

We compare the performance of the static method with the random method as the baseline. Fig. 6.2 shows the result in different configurations. It can be observed that 1) the static method significantly outperform the random method; and 2) there is an error distance rate of more than 5% for the static method in most of the configurations, which is because of the unstable problem we will explain later.

Algorithm 7 Training Threshold in Static Method

Input: A sequence of N queries together with estimated revenue for their ads $X = X_1, X_2, ...X_N$
k : the maximum number of ads to select
Algorithm:

```
ArrayList list
for i = 0 to N-1 do
  list.add(X_{i3})
  list.add(X_{i2})
  list.add(X_{i1})
end for
sortList = sortDescendOrder(list)
Threshold = sortList[k − 1]
```

Output: Threshold to determine whether to impress an ad

6.6.3 Unstable Problem of the Static Method

From the experimental results, we find that although the static method has obtained competitive performance, there is still around 5% distance from the optimal result. It seems that 5% is a very marginal distance; however, in sponsored search, such a marginal distance means billions of US dollars per year for search engine companies from the statistics in eMarketer. Thus to decrease this distance is an important research issue.

The main reason for this distance is that the data has unstable property. In the static method, it is assumed that the distribution of the estimated revenue is stable over times. But this assumption is not accurate. In fact, the revenue distribution is changing over times. To verify this, we make statistics on the dataset about the change of different query windows.

First, we calculate the average estimated revenue for all queries in each window and make statistics on the distance between the current window and its previous window. Fig. 6.3 shows the distribution of these distances (the left is original and the right is the one after normalization). Secondly, we record the threshold learned in each window and make statistics on the distance be-

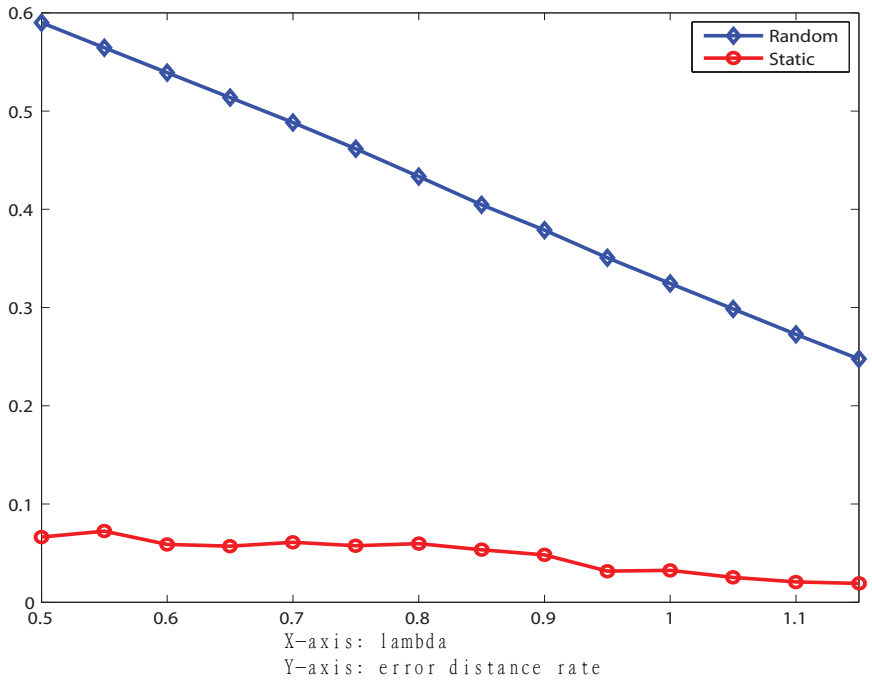


Figure 6.2: Performance of static method

tween current window and its previous window. Fig. 6.4 shows the distribution of these distances (the left is original and the right is the one after normalization). It can be observed, in both figures, more then 10% of the cases, the data has changed for more than 10%. This demonstrates that the data has the unstable problem.

Some may argue that the statistics on the estimated revenue are correlated with specific revenue estimation method and may not generalizable in other cases. Thus we study more insight for the unstable problem. We verify the change of data from two aspects, 1) the change of query type distribution and 2) the change for CTR. In the former, we cluster all the queries into 1,000 groups based on the term frequency. Within each window, we calculate the probability for a query to different groups. Then we make statistics on distribution of the KL distances between current window and its previous window. Fig. 6.5 shows the

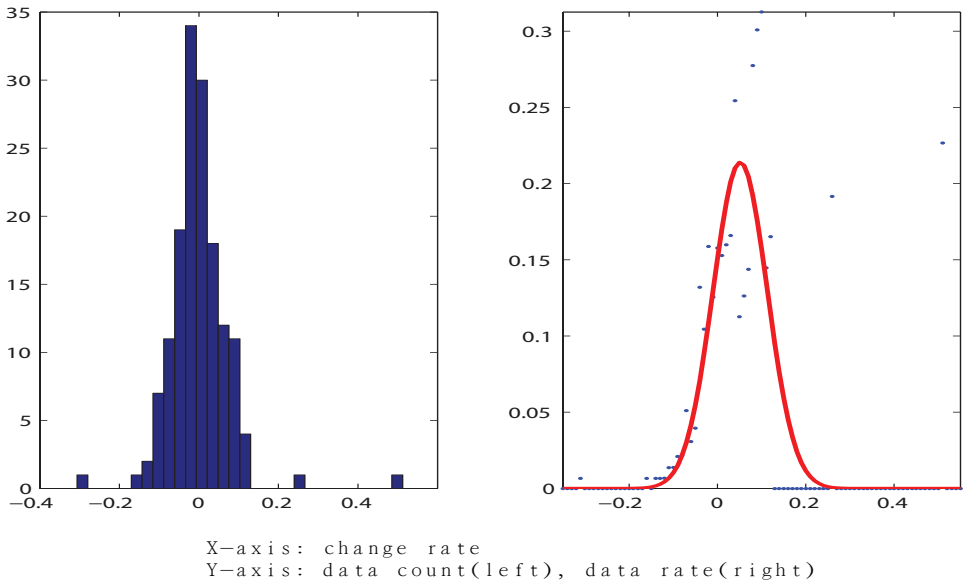


Figure 6.3: Distribution of the changed average revenue

result. In the latter, we estimate the CTR for each group and calculate the average change between current window and previous window. Fig. 6.6 shows the distributions of the change. Both figures have shown that the data is changing at a certain rate. Therefore the change will result in the unstable property of estimated revenue values as the insight reasons.

The unstable problem is a main challenge for the issue. Using static method means that more than 5% of total revenue would be lost for search engine companies. Thus a dynamic algorithm is desired to adapt to the change when deciding the strategy in impressing the ads.

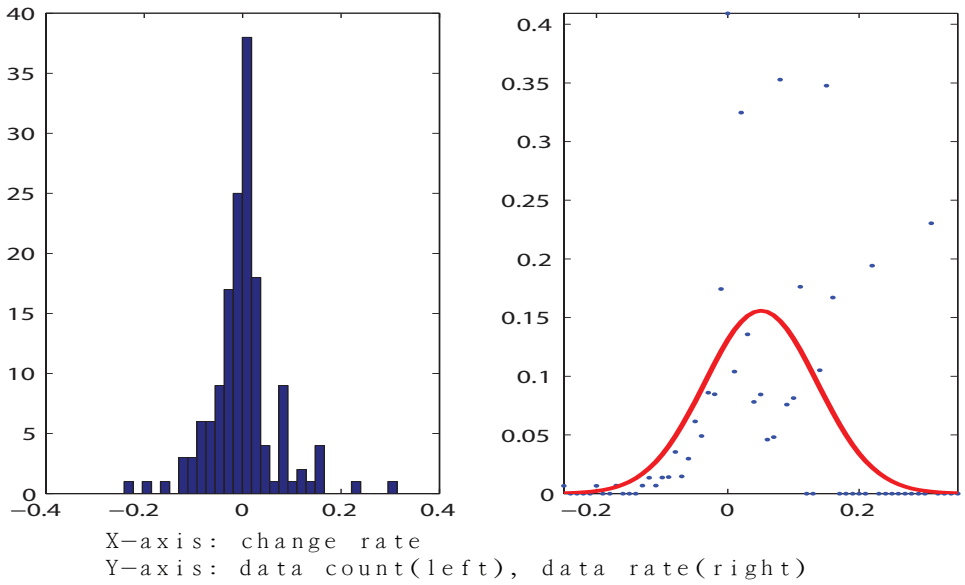


Figure 6.4: Distribution of the changed threshold

6.7 Proposed Dynamic Method for Impression Efficiency Optimization

6.7.1 Proposed Dynamic Method

To solve the unstable problem, we explore the methods in the framework of the secretary problem. As mentioned in previous section, our task is to solve the constrained 3-tuples multi-choice problem, which has never been investigated before. The most similar problem with us is unconstrained 1-tuple multi-choice problem. Previous work in [74] has proposed a recursive algorithm. The expected performance of this algorithm can achieve $(1 - O(1/\sqrt{k}))v$, where v is the maximum value and k is the number of elements to be selected. Here we propose a novel algorithm for our task in this work as an extension.

The proposed algorithm is shown in Algorithm 8 and Fig. 6.7. If $k = 1$, we observe the first N/e queries without ads impression, and set the largest value observed as the threshold. Then for the rest of the queries, we select the first ad that has the estimated

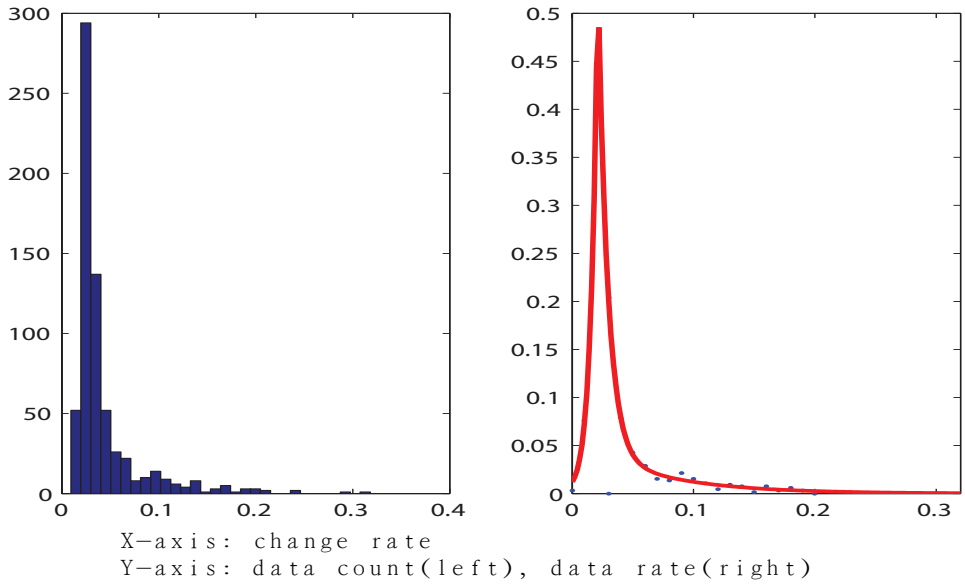


Figure 6.5: The change of query type

value larger than this threshold. If $k > 1$, we first sample an m from binomial distribution $B(N, 0.5)$, and then recursively to select $k/2$ ads from the m queries. Set the $k/2^{th}$ largest value in m queris as the threshold for remained $N - m$ queries. Then select ads that is larger than the threshold under the ranking constraints.

6.7.2 Empirical Study of the Dynamic Method

We show the real competitive ratio in different configurations of λ, as shown in Fig. 6.8. From the figure, we can observe the competitive ratio is over 0.97 in all the configurations.

Fig. 6.9 shows the experimental result of the dynamic method compared with the static method. It can be observed that the dynamic algorithm can consistently outperform the static method with significant improvement. Table 6.3 shows the improvement in real revenue. This demonstrate that the dynamic algorithm we proposed is effective in solving the unstable problem.

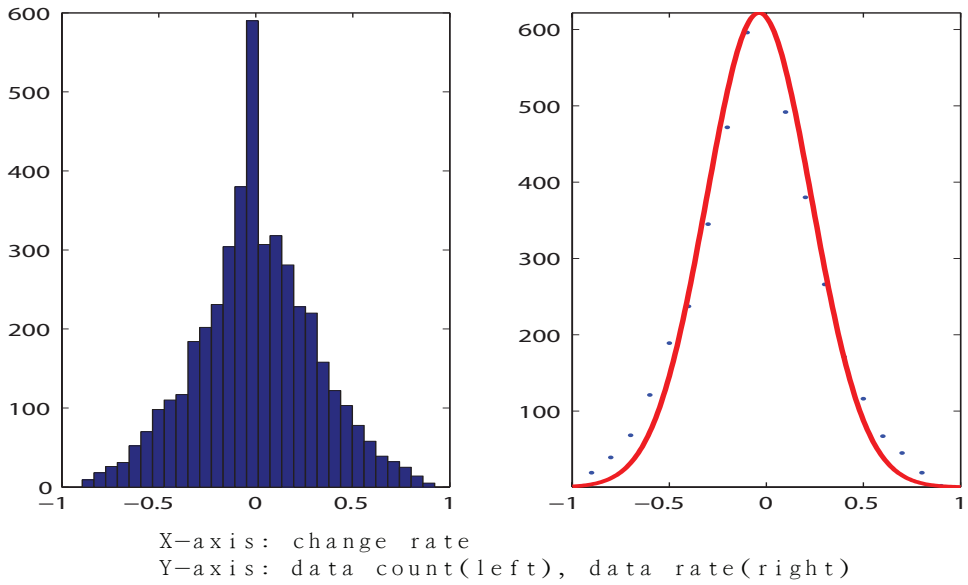


Figure 6.6: The change of click-through rate

Table 6.3: Improvement of the dynamic algorithm compared with the static algorithm

Rate	0.5	0.55	0.60	0.65	0.70	0.75	0.80
Imprv. (%)	8.887	13.63	12.02	13.34	14.15	17.6	21.82
Rate	0.85	0.90	0.95	1.00	1.05	1.10	1.15
Imprv. (%)	20.22	21.65	17.26	20.71	20.3	16.58	17.1

6.8 Combination of Static and Dynamic Methods

6.8.1 Combination Approach

In previous sections, we have tried both static and dynamic methods for the problem. The advantage of the static method is that it utilizes history information, but it cannot adapt to the change; while the dynamic method can adapt well to the change, but it does not utilize the help from the history data. Thus it is natural to combine these two kinds of methods to improve the performance. Therefore, in this section, we propose a heuristic method to combine static and dynamic methods to-

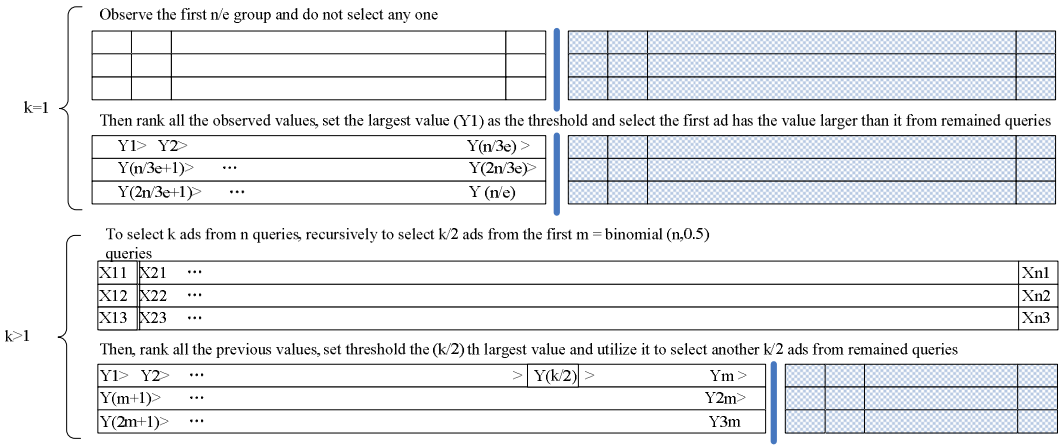


Figure 6.7: Dynamic method illustration

gether. The main idea is as follows. We set a lower bound and upper bound for the selection in dynamic method. In case that the estimated revenue of an ad is below the lower bound, it should not be selected; and similarly, in case that it is beyond the an upper bound, it will be selected without comparing to current threshold value. The lower bound and upper bound are defined according to the history information calculated in the static method. In this way, the dynamic method adapts to change in a range constrained by the static method for performance improvement. Algorithm 9 shows more details for the combination. The time complexity is mainly from the sorting, which is $O(n \log n)$.

6.8.2 Experimental Verification

Fig. 6.10 shows the experimental results of the comparison of the combination method with dynamic method. It can be demonstrated that the combination method consistently performs the best in all configurations. The improvement of the real revenue is shown in Table 6.4. The combination can well solve the unstable problem identified before. In sponsored search community, such improvement means a significant increase of revenue for

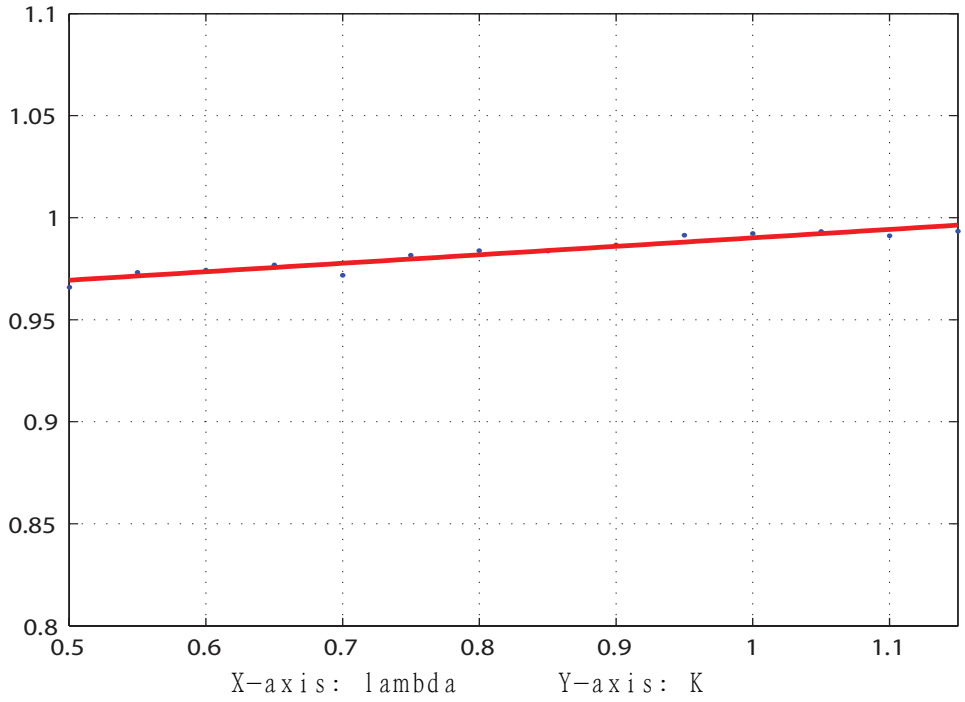


Figure 6.8: Competitive ratio on different K

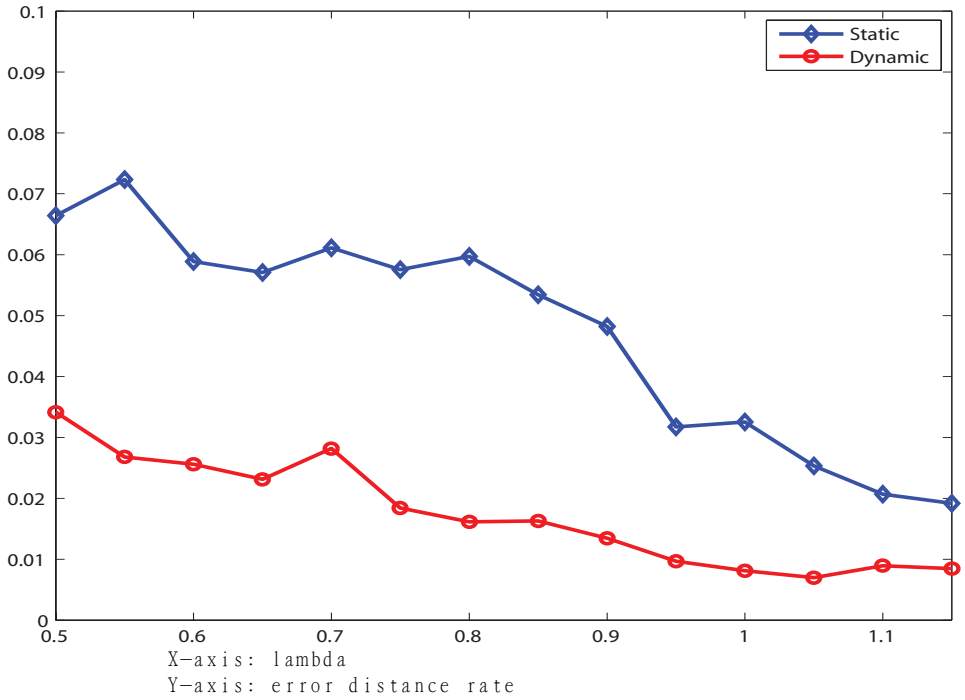


Figure 6.9: Performance of dynamic method

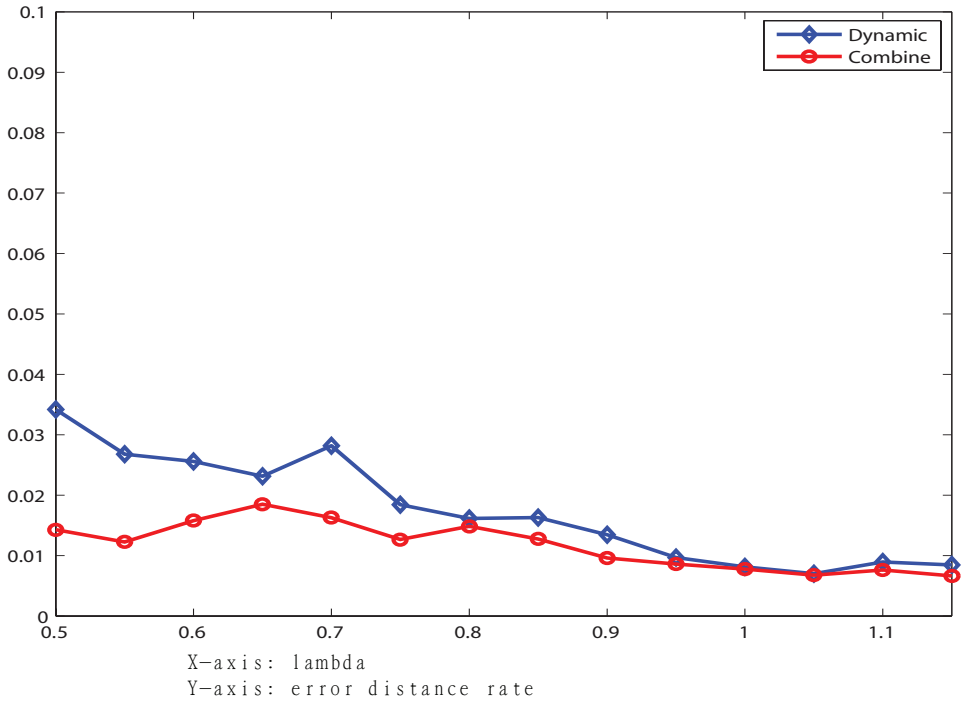


Figure 6.10: Performance of combination method

Table 6.4: Improvement of the combination algorithm compared with the dynamic algorithm

Rate	0.5	0.55	0.60	0.65	0.70	0.75	0.80
Imprv. (%)	6.289	5.503	3.799	0.6202	4.165	3.534	0.0572
Rate	0.85	0.90	0.95	1.00	1.05	1.10	1.15
Imprv. (%)	1.482	2.703	0.5186	1.084	0.1678	1.025	1.070

search engine companies. Fig. 6.11 shows the performance of four methods in real revenue for demonstration in practical application. It can be observed that the algorithms proposed in this work is practical in real applications.

6.9 Summary

In this work, we formulate the problem of impression efficiency optimization in sponsored search under the secretary problem framework. Through experiments on real world dataset, we

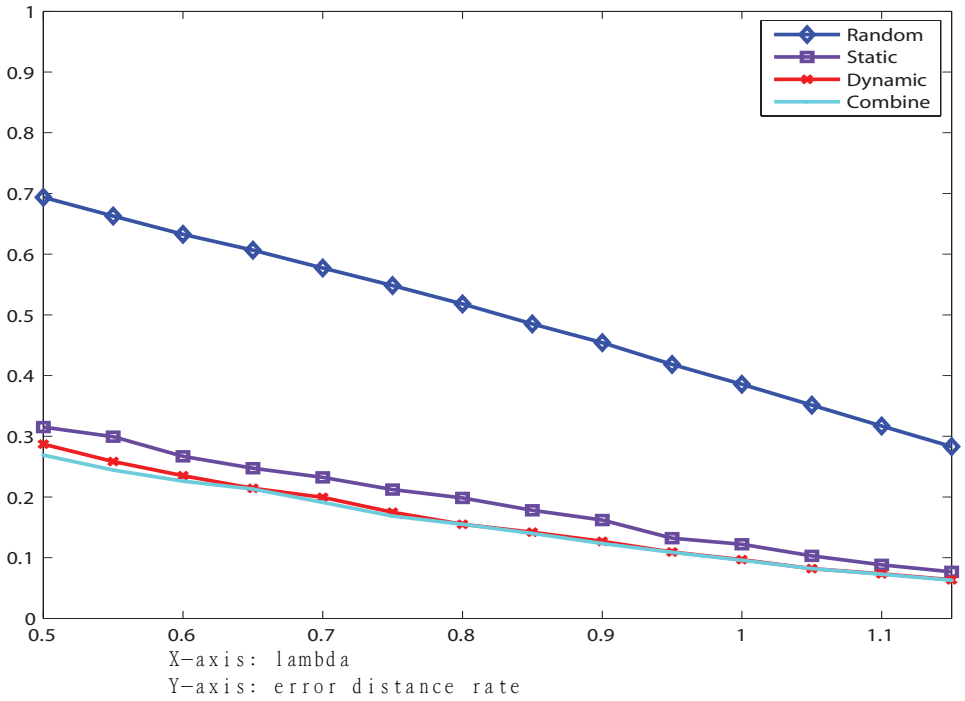


Figure 6.11: Performance in real revenue case

found that the data in sponsored search is unstable over time and direct static methods cannot achieve good performance. A dynamic algorithm is proposed to solve this problem. By experimental verification, our algorithm has a significant improvement from static methods. At last, we combine the history information in static methods into the dynamic method using heuristic method, experimental results show that another significant improvement can be obtained.

Algorithm 8 Dynamic Method

Input: A sequence of N queries together with estimated revenue for their ads $X = X_1, X_2, ...X_N$
k : the maximum number of ads to select
Algorithm:

```
ArrayList mList, kList, valueList
double threshold
mList.add(N), kList.add(k)
while k! = 1 do
  N = Binormial(N, 0.5), k = k/2.
  mList.add(N), kList.add(k).
end while
for g = mList.length − 1 to 0 do
  if g equals to mList.length − 1 then
    Observe from X_1 to X_{mList.get(g)/e}
    Put X_ij into valueList
    Select the largest X_ij as threshold
    for i = X_{mList.get(g)/e+1} to X_{mList.get(g)} do
      Put X_ij into valueList
      Impress ads to users if the value is larger than the threshold as in
      Algorithm 6 until the impression number reaches kList.get(g).
    end for
  else
    threshold = the (kList.get(g)/2)^th largest value in valueList
    for i = X_{mList.get(g−1)+1} to X_{mList.get(g)} do
      Put X_ij into valueList
      Impress ads to users if the value is larger than the threshold as in
      Algorithm 6 until the impression number reaches kList.get(g).
    end for
  end if
end for
```

Output: The strategy for impressing ads for current N queries.

Algorithm 9 Combination Method

Input: A sequence of N queries
threshold: the threshold in previous window
margin: to decide the up-bound and low-bound
Algorithm:

```
lowBound = threshold-margin;
upBound = threshold+margin
mList.add(N), kList.add(k)
while k! = 1 do
  N = Binormial(N, 0.5), k = k/2.
  mList.add(N), kList.add(k).
end while
for g = mList.length − 1 to 0 do
  if g equals to mList.length − 1 then
    Observe from X_1 to X_{mList.get(g)/e}
    Impress ads if the value is larger than upBound
    Select the largest X_ij as the threshold
    threshold = Min(threshold, lowBound)
    for i = X_{mList.get(g)/e+1} to X_{mList.get(g)} do
      Impress ads to users as in Algorithm 6 if the value is larger than
      upBound or if kList.get(g) is larger than selected ads number and
      the value is larger than the threshold
    end for
  else
    threshold = the (kList.get(g)/2)^th largest value in valueList
    threshold = Min(threshold, lowBound)
    for i = X_{mList.get(g−1)+1} to X_{mList.get(g)} do
      Impress ads to users as in Algorithm 6 if the value is larger than
      upBound or if kList.get(g) is larger than selected ads number and
      the value is larger than the threshold
    end for
  end if
end for
```

Output: The strategy for impressing ads

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The thesis investigates effective fusion-based approaches for recommender systems. Specifically, the target is to solve four limitations from the four different levels in the evaluation structure of recommender systems as shown in Fig. 1.2: (1) the relational dependency is ignored in previous fusion methods for single measure and dimension. (2) single-measure-adapted algorithms cannot adapt to multi-measure performance; (3) single-dimension-adapted algorithms cannot adapt to multi-dimensional performance; and (4) impression efficiency optimization in recommendation is not carefully considered.

For the first limitation, we propose a relational fusion approach, based on extended multi-scale continuous conditional random fields (CRF). The approach can model the relational dependency by Markov property and it is designed to integrate multiple features. Experimental results demonstrate that the relational dependency is effective in improving the performance of recommender systems and CRF framework is effective in combining multiple features.

For the second limitation, we propose methods to fuse regression-oriented and ranking-oriented algorithms for multi-measure adaption in recommender systems. We propose fusion approaches for

both model-based and memory-based CF methods. In model-based methods, we propose to combine the objective functions of two competitive methods, regression-adapted probabilistic matrix factorization (PMF) [128] and ranking-adapted list-wise matrix factorization (LMF) [139]; and in memory-based methods, we propose to combine the results of two competitive methods, regression-adapted user-based Pearson correlation coefficient (PCC) [17] and ranking-adapted EigenRank [91]. Experimental results verify that the combination is effective in improving performances on both regression and ranking measurements.

For the third limitation, we propose methods to fuse quality-based and relevance-based algorithms as a preliminary study for multi-dimensional recommendation. We propose an effective fusion approach, which enables principled and natural integration with features derived from both quality-based and relevance-based algorithms. Experimental results identify that the combination is effective in improving the performance in the integrated metric significantly and the fusion approach outperforms fundamental combination methods.

For the last limitation, we formulate the problem and propose a dynamic algorithm to optimize the impression efficiency in sponsored advertisements recommendation. We identify the unstable problem for fundamental static methods by statistical data analysis. Through empirical study, we show that the dynamic algorithm is effective in solving the unstable problem.

We believe the improvements proposed in this thesis are worthwhile in practice. Users would benefit from convenient and accurate recommendation service with less intrusion; and E-business companies would obtain more revenue from more commercial behaviors of the Web users.

7.2 Future Work

Although CF algorithms, the key technique in recommender system, have been investigated deeply over the decades, there are still many challenging problems in need to solve. As future work, we will still focus on improvement from the work proposed in this thesis.

For the first work on relational fusion based on CRF, the limitations lie in the computation complexity. Although the inference process based on Gibbs sampling has reduced the complexity from exponential complexity to linear complexity, and it can be applied in large scale dataset, the converging speed is still slow comparing to traditional model-based algorithms, such as PMF. Thus the future work is to explore more effective optimization methods to speed up the inference process.

For the second work on the fusion of regression-oriented and ranking-oriented CF algorithms, the limitation is that the combination depends on the formulations of selected algorithms. Thus for other kind of algorithms, it might has problem in generalization. Thus as future work, we will explore how to build a general framework for the combination of all kinds of regression-oriented and ranking-oriented CF algorithms.

For the third work on the fusion of quality-based and relevance-based algorithms in multi-dimensional recommendation, it is still a preliminary work. It only considers two dimensions, and there are many more dimensions that need to explore, such as diversity, coverage, etc. Thus as future work, we will explore how to consider more dimensions in recommendation.

For the last work on optimizing the impression efficiency in sponsored search, the limitation is that the impression efficiency optimization and the advertisement ranking are treated as two separate tasks as a preliminary work. If the two tasks are jointly optimized, it would make more improvement, which is also as

the future work of this thesis.

In the first work, we investigate the relational recommendation framework to combine multiple features for the traditional regression problem. However, in the other work, it is still a primary study. Relational dependency is not utilized in the combination. Thus in future work, we plan to extend the previous work to utilize the relational dependency information in order to improve the performance of recommender systems.

□ **End of chapter.**

Appendix A

Publications

A.1 Publications in Ph. D study

- Xin Xin, Michael R. Lyu, and Irwin King. CMAP: Effective Fusion of Quality and Relevance for Multi-criteria Recommendation (Full Paper). In Proceedings of ACM 4th International Conference on Web Search and Data Mining (WSDM 2011), Hong Kong, February 2011.
- Xin Xin, Irwin King, Hongbo Deng, and Michael R. Lyu. A Social Recommendation Framework Based on Multi-scale Continuous Conditional Random Fields (Full and Oral Paper). In Proceedings of ACM 18th Conference on Information and Knowledge Management (CIKM 2009), Hong Kong, November 2009.

A.2 Publications in Master study

- Xin Xin, Juanzi Li, Jie Tang, and Qiong Luo. Academic Conference Homepage Understanding Using Hierarchical Conditional Random Fields (Full and Oral Paper). In Proceedings of ACM 17th Conference on Information and Knowledge Management (CIKM 2008), Napa Valley, CA, October 2008.

- Xin Xin, Juanzi Li, and Jie Tang. Enhancing Semantic Web by Semantic Annotation: Experiences in Building an Automatic Conference Calendar (Short Paper). In Proceedings of the 2007 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2007), Fremont, CA, November 2007.

A.3 Unpublished Work in Ph. D study

- Xin Xin, Haiqin Yang, Michael R. Lyu, and Irwin King. Combining Regression and Ranking in Collaborative Filtering. Submitted to CIKM 2011.
- Xin Xin, Wei Wang, Wei Yu, Jie Tang, Irwin King and Michael R. Lyu. Learning to Impress in Sponsored Search. Preparing to submit to WWW 2012.
- Wei Wang, Xin Xin, Irwin King, Jie Tang, and Michael R. Lyu. Compete or Collaborate? Incorporating Relational Influence within Search Results into Click Model in Sponsored Search. Submitted to CIKM 2011.

□ **End of chapter.**

Bibliography

[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 Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[2] D. Agarwal and B. Chen. Regression-based latent factor models. In *Proc. of SIGKDD'09*, pages 19–28. ACM, 2009.

[3] D. Agarwal, B. Chen, and P. Elango. Fast online learning through offline initialization for time-sensitive recommendation. In *Proc. of SIGKDD'10*, pages 703–712. ACM, 2010.

[4] X. Amatriain, N. Lathia, J. Pujol, H. Kwak, and N. Oliver. The wisdom of the few: a collaborative filtering approach based on expert opinions from the web. In *Proc. of SIGIR'09*, pages 532–539. ACM, 2009.

[5] R. Andersen, C. Borgs, J. Chayes, U. Feige, A. Flaxman, A. Kalai, V. Mirrokni, and M. Tennenholtz. Trust-based recommendation systems: an axiomatic approach. In *Proc. of WWW'08*, pages 199–208, New York, NY, USA, 2008. ACM.

[6] W. Anderson. *Continuous-time Markov chains: applications-oriented approach*. Springer, 1991.

[7] C. Andrieu, N. De Freitas, A. Doucet, and M. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43, 2003.

[8] N. Archak, V. Mirrokni, and S. Muthukrishnan. Mining advertiser-specific user behavior using adfactors. In *Proc. of WWW'10*, pages 31–40. ACM, 2010.

[9] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. Online auctions and generalized secretary problems. *ACM SIGecom Exchanges*, 7(2):1–11, 2008.

[10] M. Balabanović and Y. Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[11] H. Bao and E. Chang. AdHeat: an influence-based diffusion model for propagating hints to match ads. In *Proc. of WWW'10*, pages 71–80. ACM, 2010.

[12] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: using social and content-based information in recommendation. In *Proc. of the NCAI'98*, pages 714–720, 1998.

[13] P. Bedi, H. Kaur, and S. Marwaha. Trust based recommender system for the semantic web. In *Proc. of IJCAI'07*, pages 2677–2682, 2007.

[14] R. Bell, Y. Koren, and C. Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proc. of SIGKDD'07*, pages 95–104. ACM, 2007.

[15] M. Bendersky, E. Gabrilovich, V. Josifovski, and D. Metzler. The anatomy of an ad: structured indexing and re-

trieval for sponsored search. In *Proc. of WWW'10*, pages 101–110. ACM, 2010.

[16] D. Billsus and M. Pazzani. Learning collaborative information filters. In *Proc. of ICML'98*, volume 54, 1998.

[17] 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.

[18] A. Broder, M. Ciaramita, M. Fontoura, E. Gabrilovich, V. Josifovski, D. Metzler, V. Murdock, and V. Plachouras. To swing or not to swing: learning when (not) to advertise. In *Proc. of CIKM'08*, pages 1003–1012. ACM, 2008.

[19] A. Broder, P. Ciccolo, E. Gabrilovich, V. Josifovski, D. Metzler, L. Riedel, and J. Yuan. Online expansion of rare queries for sponsored search. In *Proc. of WWW'09*, pages 511–520. ACM, 2009.

[20] R. Burke. Hybrid recommender systems: survey and experiments. In *Proc. of UAI'02*, volume 12, pages 331–370. Springer, 2002.

[21] G. Buscher, S. Dumais, and E. Cutrell. The good, the bad, and the random: an eye-tracking study of ad quality in web search. In *Proc. of SIGIR'10*, pages 42–49. ACM, 2010.

[22] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proc. of the ICML'07*, pages 129–136. ACM, 2007.

[23] W. Chen, J. Chu, J. Luan, H. Bai, Y. Wang, and E. Chang. Collaborative filtering for orkut communities: discovery of user latent behavior. In *Proc. of WWW'09*, pages 681–690. ACM, 2009.

[24] W. Chen, D. Zhang, and E. Chang. Combinational collaborative filtering for personalized community recommendation. In *Proc. of SIGKDD'08*, pages 115–123. ACM, 2008.

[25] Y. Chien and E. George. A bayesian model for collaborative filtering. In *Proc. of the Seventh International Workshop on Artificial Intelligence and Statistics*, 1999.

[26] Y. Choi, M. Fontoura, E. Gabrilovich, V. Josifovski, M. Mediano, and B. Pang. Using landing pages for sponsored search ad selection. In *Proc. of WWW'10*, pages 251–260. ACM, 2010.

[27] W. Chu and S. Park. Personalized recommendation on dynamic content using predictive bilinear models. In *Proc. of WWW'09*, pages 691–700. ACM, 2009.

[28] F. Chua and E. Lim. Trust network inference for online rating data using generative models. In *Proc. of SIGKDD'10*, pages 889–898. ACM, 2010.

[29] M. Ciaramita, V. Murdock, and V. Plachouras. Online learning from click data for sponsored search. In *Proc. of WWW'08*, pages 227–236. ACM, 2008.

[30] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proc. of ACM SIGIR Workshop on Recommender Systems*, 1999.

[31] F. Constantin, J. Feldman, S. Muthukrishnan, and M. Pál. An online mechanism for ad slot reservations with cancellations. In *Proc. of SIAM'09*, pages 1265–1274. Society for Industrial and Applied Mathematics, 2009.

[32] B. Croft, D. Metzler, and T. Strohman. *Search engines: information retrieval in practice*. Addison Wesley, 2009.

[33] C. Danescu-Niculescu-Mizil, A. Broder, E. Gabrilovich, V. Josifovski, and B. Pang. Competing for users' attention: on the interplay between organic and sponsored search results. In *Proc. of WWW'10*, pages 291–300. ACM, 2010.

[34] S. Daruru, N. Marin, M. Walker, and J. Ghosh. Pervasive parallelism in data mining: dataflow solution to co-clustering large and sparse netflix data. In *Proc. of SIGKDD'09*, pages 1115–1124. ACM, 2009.

[35] K. Dembczynski, W. Kotlowski, and D. Weiss. Predicting ads' click-through rate with decision rules. In *Proc. of WWW'08*. Citeseer, 2008.

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

[37] C. Ding, H. Simon, R. Jin, and T. Li. A learning framework using green's function and kernel regularization with application to recommender system. In *Proc. of SIGKDD'07*, pages 260–269. ACM, 2007.

[38] J. Duchi, L. Mackey, and M. Jordan. On the consistency of ranking algorithms. In *Proc. of ICML'10*. Citeseer, 2010.

[39] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, November 2000.

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

[41] E. Dynkin. Optimal choice of the stopping moment of a Markov process. In *Dokl. Akad. Nauk SSSR*, volume 150, 1963.

[42] A. Elberse. Should you invest in the long tail? *Harvard Business Review*, 86(7/8):88–96, 2008.

[43] 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.

[44] C. Elkan. Log-linear models and conditional random fields. In *Tutorial notes at CIKM'08*, 2008.

[45] T. Ferguson. Who solved the secretary problem? *Statistical Science*, 4(3):282–289, 1989.

[46] D. Fleder and K. Hosanagar. Recommender systems and their impact on sales diversity. In *Proc. of EC'2007*, pages 192–199. ACM, 2007.

[47] 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.

[48] P. Freeman. The secretary problem and its extensions: a review. *International Statistical Review/Revue Internationale de Statistique*, 51(2):189–206, 1983.

[49] M. Gardner. Mathematical games. *Scientific American*, 202(3):172–182.

[50] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. Pazzani. An energy-efficient mobile recommender system. In *Proc. of SIGKDD'10*, pages 899–908. ACM, 2010.

[51] L. Getoor and M. Sahami. Using probabilistic relational models for collaborative filtering. In *Proc. of WEBKDD'99)*, 1999.

[52] A. Ghose and S. Yang. Analyzing search engine advertising: firm behavior and cross-selling in electronic markets. In *Proc. of WWW'08*, pages 219–226. ACM, 2008.

[53] J. Golbeck. Generating predictive movie recommendations from trust in social networks, 2006.

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

[55] T. Graepel, J. Candela, T. Borchert, and R. Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in Microsofts Bing search engine. In *Proc. of ICML'10*. Citeseer, 2010.

[56] 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.

[57] Z. Guan, C. Wang, J. Bu, C. Chen, K. Yang, D. Cai, and X. He. Document recommendation in social tagging services. In *Proc. of WWW'10*, pages 391–400. ACM, 2010.

[58] F. Guo, C. Liu, A. Kannan, T. Minka, M. Taylor, Y. Wang, and C. Faloutsos. Click chain model in web search. In *Proc. of WWW'09*, pages 11–20. ACM, 2009.

[59] I. Guy, N. Zwerdling, I. Ronen, D. Carmel, and E. Uziel. Social media recommendation based on people and tags. In *Proc. of SIGIR'10*, pages 194–201. ACM, 2010.

[60] J. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. *Unpublished manuscript*, 1971.

[61] A. Harpale and Y. Yang. Personalized active learning for collaborative filtering. In *Proc. of SIGIR'08*, pages 91–98. ACM, 2008.

[62] Q. He, J. Pei, D. Kifer, P. Mitra, and L. Giles. Context-aware citation recommendation. In *Proc. of WWW'10*, pages 421–430. ACM, 2010.

[63] X. He, R. S. Zemel, and M. A. Carreira-Perpinan. Multiscale conditional random fields for image labeling. In *Proc. of CVPR'04*, volume 2, pages 695–702, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[64] 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.

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

[66] W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proc. of SIGCHI'95*, pages 194–201. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 1995.

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

[68] M. Jahrer, A. Toscher, and R. Legenstein. Combining predictions for accurate recommender systems. In *Proc. of SIGKDD'10*, pages 693–702. ACM, 2010.

[69] M. Jamali and M. Ester. Trustwalker: a random walk model for combining trust-based and item-based recommendation. In *Proc. of SIGKDD'09*, pages 397–406. ACM, 2009.

[70] M. Jamali, G. Haffari, and M. Ester. Modeling the temporal dynamics of social rating networks using bidirectional effects of social relations and rating patterns. In *Proc. of WWW'11*, pages 527–536. ACM, 2011.

[71] B. Jansen and M. Resnick. An examination of searcher's perceptions of nonsponsored and sponsored links during ecommerce Web searching. *Journal of the American Society for Information Science and Technology*, 57(14):1949–1949, 2006.

[72] N. Kawamae. Serendipitous recommendations via innovators. In *Proc. of SIGIR'10*, pages 218–225. ACM, 2010.

[73] B. Kégl and R. Busa-Fekete. Boosting products of base classifiers. In *Proc. of ICML'09*, pages 497–504. ACM, 2009.

[74] R. Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *Proc. of SIAM'05*, pages 630–631. Society for Industrial and Applied Mathematics, 2005.

[75] I. Konstas, V. Stathopoulos, and J. Jose. On social networks and collaborative recommendation. In *Proc. of SIGIR'09*, pages 195–202. ACM, 2009.

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

[77] Y. Koren. Collaborative filtering with temporal dynamics. In *Proc.s of SIGKDD'10*, pages 89–97. ACM, 2010.

[78] T. T. Kristjansson, A. Culotta, P. A. Viola, and A. Mccallum. Interactive information extraction with constrained

conditional random fields. In *Proc. of AAAI'04*, pages 412–418, 2004.

[79] S. Kullback and R. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[80] 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. of KDD Cup and Workshop*, volume 23, pages 720–727, 2007.

[81] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML'01*, pages 282–289, 2001.

[82] K. Lang. Newsweeder: learning to filter netnews. In *Proc. of ML'95*, 1995.

[83] N. Lathia, S. Hailes, L. Capra, and X. Amatriain. Temporal diversity in recommender systems. In *Proc. of SIGIR'10*, pages 210–217, 2010.

[84] N. Lawrence and R. Urtasun. Non-linear matrix factorization with gaussian processes. In *Proc. of ICML'09*, pages 601–608. ACM, 2009.

[85] B. Li, Q. Yang, and X. Xue. Transfer learning for collaborative filtering via a rating-matrix generative model. In *Proc. of ICML'09*, pages 617–624. ACM, 2009.

[86] L. Li, W. Chu, J. Langford, and R. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proc. of WWW'10*, pages 661–670. ACM, 2010.

[87] S. Li. Markov random field models in computer vision. *Lecture Notes in Computer Science*, 1994.

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

[89] C. Liu, H. Yang, J. Fan, L. He, and Y. Wang. Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce. In *Proc. of WWW'10*, pages 681–690. ACM, 2010.

[90] J. Liu. *Monte Carlo strategies in scientific computing*. Springer, 2001.

[91] 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.

[92] N. Liu, M. Zhao, and Q. Yang. Probabilistic latent preference analysis for collaborative filtering. In *Proc. of CIKM'09*, pages 759–766. ACM, 2009.

[93] T. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[94] T. Lu and C. Boutilier. Learning mallows models with pairwise preferences. In *Proc. of ICML'11*, 2011.

[95] Y. Lv, T. Moon, P. Kolari, Z. Zheng, X. Wang, and Y. Chang. Learning to model relatedness for news recommendation. In *Proc. of WWW'11*, pages 57–66. ACM, 2011.

[96] 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.

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

[98] L. Mackey, D. Weiss, and M. Jordan. Mixed membership matrix factorization. In *Proc. ICML'10*. Citeseer, 2010.

[99] L. Marable. False oracles: consumer reaction to learning the truth about how search engines work, results of an ethnographic study. 2003.

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

[101] B. Marlin. Modeling user rating profiles for collaborative filtering. In *Proc. of NIPS'04*, volume 16, pages 627–634. MIT Press, 2004.

[102] F. McSherry and I. Mironov. Differentially private recommender systems: building privacy into the net. In *Proc. SIGKDD'09*, pages 627–636. ACM, 2009.

[103] B. Mehta and W. Nejdl. Attack resistant collaborative filtering. In *Proc. of SIGIR'08*, pages 75–82. ACM, 2008.

[104] P. Melville, R. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Proc. of UAI'02*, pages 187–192. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002.

[105] B. Miller, I. Albert, S. Lam, J. Konstan, and J. Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *Proc. of the 8th International Conference on Intelligent User Interfaces*, pages 263–266. ACM New York, NY, USA, 2003.

[106] R. Mooney, P. Bennett, and L. Roy. Book recommending using text categorization with extracted information. In *Recommender Systems. Papers from 1998 Workshop. Technical Report WS-98*, volume 8, 1998.

[107] R. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proc. DL'00*, pages 195–204. ACM New York, NY, USA, 2000.

[108] R. Nakamoto, S. Nakajima, J. Miyazaki, S. Uemura, and H. Kato. Investigation of the effectiveness of tag-based contextual collaborative filtering in website recommendation. *Advances in Communication Systems and Electrical Engineering*, pages 309–318, 2008.

[109] N. Nanas, M. Vavalis, and A. De Roeck. A network-based model for high-dimensional information filtering. In *Proc. of SIGIR'10*, pages 202–209. ACM, 2010.

[110] J. O'Donovan and B. Smyth. Trust in recommender systems. In *Proc. of IUI'05*, pages 167–174, New York, NY, USA, 2005. ACM.

[111] K. Onuma, H. Tong, and C. Faloutsos. Tangent: a novel,'surprise me', recommendation algorithm. In *Proc. of SIGKDD'09*, pages 657–666. ACM, 2009.

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

[113] R. Pan and M. Scholz. Mind the gaps: weighting the unknown in large-scale one-class collaborative filtering. In *Proc. of SIGKDD'09*, pages 667–676. ACM, 2009.

[114] S. Pandey, K. Punera, M. Fontoura, and V. Josifovski. Estimating advertisability of tail queries for sponsored search. In *Proc. of SIGIR'10*, pages 563–570. ACM, 2010.

[115] D. Pavlov and D. Pennock. A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains. In *Proc. of NIPS'03*, pages 1465–1472. MIT; 1998, 2003.

[116] M. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5):393–408, 1999.

[117] M. Pazzani and D. Billsus. Learning and revising user profiles: the identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.

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

[119] T. Qin, T. Liu, X. Zhang, D. Wang, and H. Li. Global ranking using continuous conditional random fields. In *Proc. of NIPS'08*, 2008.

[120] F. Radlinski, A. Broder, P. Ciccolo, E. Gabrilovich, V. Josifovski, and L. Riedel. Optimizing relevance and revenue in ad search: a query substitution approach. In *Proc. of SIGIR'09*, pages 403–410. ACM, 2008.

[121] S. Rendle, L. Balby Marinho, A. Nanopoulos, and L. Schmidt-Thieme. Learning optimal ranking with tensor factorization for tag recommendation. In *Proc. of SIGKDD'09*, pages 727–736. ACM, 2009.

[122] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proc. of WWW'10*, pages 811–820. ACM, 2010.

[123] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proc. of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

[124] E. Rich. User modeling via stereotypes. *Readings in Intelligent User Interfaces*, pages 329–341, 1998.

[125] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *Proc. of WWW'07*, pages 521–530. ACM, 2007.

[126] S. Robertson and S. Walker. Threshold setting in adaptive filtering. *Journal of Documentation*, 56(3):312–331, 2000.

[127] S. Rosset, C. Perlich, and Y. Liu. Making the most of your data: Kdd cup 2007 how many ratings winner's report. *ACM SIGKDD Explorations Newsletter*, 9(2):66–69, 2007.

[128] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Proc. of NIPS'08*, volume 20, pages 1257–1264. Citeseer, 2008.

[129] R. Salakhutdinov and N. Srebro. Collaborative filtering in a non-uniform world: learning with the weighted trace norm. In *Proc. of NIPS'10*, 2010.

[130] 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.

[131] 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.

[132] A. Schein, A. Popescul, L. Ungar, and D. Pennock. Methods and metrics for cold-start recommendations. In *Proc. of SIGIR'06*, pages 253–260. ACM New York, NY, USA, 2002.

[133] D. Sculley. Combined regression and ranking. In *Proc. of SIGKDD'10*, pages 979–988. ACM, 2010.

[134] S. Sen, J. Vig, and J. Riedl. Tagommenders: connecting users to items through tags. In *Proc. of the WWW'09*, pages 671–680. ACM, 2009.

[135] S. Sen, J. Vig, and J. Riedl. Tagommenders: connecting users to items through tags. In *Proc. of WWW'09*, pages 671–680. ACM, 2009.

[136] S. Shalev-Shwartz, A. Gonen, and O. Shamir. Large-scale convex minimization with a low-rank constraint. In *Proc. of ICML'11*, 2011.

[137] G. Shani, D. Heckerman, and R. Brafman. An MDP-based recommender system. *Journal of Machine Learning Research*, 6(2):1265, 2006.

[138] U. Shardanand and P. Maes. Social information filtering: algorithms for automating word of mouth. In *Proc. of SIGCHI'95*, pages 210–217. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 1995.

[139] Y. Shi, M. A. Larson, and A. Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *Proc. of RecSys'10*, pages 269–272, Barcelona, Spain, 2010. ACM, ACM.

[140] L. Si and R. Jin. Flexible mixture model for collaborative filtering. In *Proc. of ICML'03*, pages 704–711. ACM, NY, USA, 2003.

[141] B. Sigurbjornsson and R. Van Zwol. Flickr tag recommendation based on collective knowledge. In *Proc. of WWW'08*, pages 327–336. ACM, 2008.

[142] A. Singh and G. Gordon. Relational learning via collective matrix factorization. In *Proc. of SIGKDD'08*, pages 650–658. ACM, 2008.

[143] I. Soboro and C. Nicholas. Combining content and collaboration in text filtering. In *Proc. of IJCAI'99*, volume 99, pages 86–91, 1999.

[144] G. Somlo and A. Howe. Adaptive lightweight text filtering. *Lecture Notes in Computer Science*, pages 319–329, 2001.

[145] Y. Song, Z. Zhuang, H. Li, Q. Zhao, J. Li, W. Lee, and C. Giles. Real-time automatic tag recommendation. In *Proc. of SIGIR'08*, pages 515–522. ACM, 2008.

[146] H. Steck. Training and testing of recommender systems on data missing not at random. In *Proc. of SIGKDD'10*, pages 713–722. ACM, 2010.

[147] D. Stern, R. Herbrich, and T. Graepel. Matchbox: large scale online bayesian recommendations. In *Proc. of WWW'09*, pages 111–120. ACM, 2009.

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

[149] J. Sueiras, A. Salafranca, and J. Florez. A classical predictive modeling approach for task who rated what? of the kdd cup 2007. *ACM SIGKDD Explorations Newsletter*, 9(2):57–61, 2007.

[150] I. Szpektor, A. Gionis, and Y. Maarek. Improving recommendation for long-tail queries via templates. In *Proc. of WWW'11*, pages 47–56. ACM, 2011.

[151] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter. Phoaks: A system for sharing recommendations. *Communications of the ACM*, 40(3):59–62, 1997.

[152] L. Ungar and D. Foster. Clustering methods for collaborative filtering. In *AAAI Workshop on Recommendation Systems*, pages 112–125, 1998.

[153] P. Viappiani and C. Boutilier. Optimal bayesian recommendation sets and myopically optimal choice query sets. In *Proc. of NIPS'10*, 2010.

[154] E. Vozalis and K. Margaritis. Analysis of recommender systems algorithms. In *Proc. of HERCMA'03*, 2003.

[155] H. Wang, Y. Liang, L. Fu, G. Xue, and Y. Yu. Efficient query expansion for advertisement search. In *Proc. of SIGIR'09*, pages 51–58. ACM, 2009.

[156] H. Wang, Y. Lu, and C. Zhai. Latent aspect rating analysis on review text data: a rating regression approach. In *Proc. SIGKDD'10*, pages 783–792. ACM, 2010.

[157] J. Wang, A. De Vries, and M. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proc. of SIGIR'06*, pages 501–508. ACM New York, NY, USA, 2006.

[158] M. Weimer, A. Karatzoglou, Q. Le, and A. Smola. Maximum margin matrix factorization for collaborative ranking. In *Proc. of NIPS'07*. MIT Press, 2007.

[159] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun. Temporal recommendation on graphs via long- and short-term preference fusion. In *Proc. SIGKDD'10*, pages 723–732. ACM, 2010.

[160] X. Xin, J. Li, J. Tang, and Q. Luo. Academic conference homepage understanding using constrained hierarchical conditional random fields. In *Proc. of CIKM'08*, pages 1301–1310, New York, NY, USA, 2008. ACM.

[161] H. Xu, E. SG, C. Caramanis, U. EDU, and S. Sanghavi. Robust matrix completion and corrupted columns.

[162] W. Xu, E. Manavoglu, and E. Cantu-Paz. Temporal click model for sponsored search. In *Proc. of SIGIR'10*, pages 106–113. ACM, 2010.

[163] 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'05*, pages 114–121. ACM New York, NY, USA, 2005.

[164] J. Yan, N. Liu, G. Wang, W. Zhang, Y. Jiang, and Z. Chen. How much can behavioral targeting help online advertising? In *Proc. of WWW'09*, pages 261–270. ACM, 2009.

[165] K. Yu, J. Lafferty, S. Zhu, and Y. Gong. Large-scale collaborative prediction using a nonparametric random effects model. In *Proc. of ICML'09*, pages 1185–1192. ACM, 2009.

[166] 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.

[167] Y. Yue, R. Patel, and H. Roehrig. Beyond position bias: examining result attractiveness as a source of presentation bias in clickthrough data. In *Proc. of WWW'10*, pages 1011–1018. ACM, 2010.

[168] Y. Zhang and J. Callan. Maximum likelihood estimation for filtering thresholds. In *Proc. of SIGIR'01*, pages 294–302. ACM New York, NY, USA, 2001.

[169] Y. Zhang, J. Callan, and T. Minka. Novelty and redundancy detection in adaptive filtering. In *Proc. of SIGIR'02*, pages 81–88. ACM New York, NY, USA, 2002.

[170] Y. Zhang and J. Koren. Efficient bayesian hierarchical user modeling for recommendation system. In *Proc. of SIGIR'07*, pages 47–54. ACM, 2007.

[171] V. Zheng, Y. Zheng, X. Xie, and Q. Yang. Collaborative location and activity recommendations with gps history data. In *Proc. of WWW'10*, pages 1029–1038. ACM, 2010.

[172] F. Zhong, D. Wang, G. Wang, W. Chen, Y. Zhang, Z. Chen, and H. Wang. Incorporating post-click behaviors into a click model. In *Proc. of SIGIR'10*, pages 355–362. ACM, 2010.

[173] D. Zhou, S. Zhu, K. Yu, X. Song, B. Tseng, H. Zha, and C. Giles. Learning multiple graphs for document recommendations. In *Proc. of WWW'08*, pages 141–150. ACM, 2008.

[174] S. Zhu, K. Yu, and Y. Gong. Stochastic relational models for large-scale dyadic data using MCMC. In *Proc. of NIPS'09*, volume 21, pages 1993–2000, 2009.

[175] X. Zhu, J. Guo, X. Cheng, P. Du, and H. Shen. A unified framework for recommending diverse and relevant queries. In *Proc. of WWW'11*, pages 37–46. ACM, 2011.

[176] Y. Zhu, G. Wang, J. Yang, D. Wang, J. Yan, J. Hu, and Z. Chen. Optimizing search engine revenue in sponsored search. In *Proc. of SIGIR'09*, pages 588–595. ACM, 2009.