September 2015

Clustering-Based Personalization

Seyed Nima Mirbakhsh
The University of Western Ontario

Supervisor
Dr. Charles X. Ling
The University of Western Ontario

Graduate Program in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree in Doctor of Philosophy

CLUSTERING-BASED PERSONALIZATION

(Thesis format: Monograph)

by

(Seyed) Nima Mirbakhsh

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor Of Philosophy

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

Acknowlegements

I would like to express my special thanks to my supervisor Professor Dr. Charles Ling, you have been a great mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. I would also like to thank my committee members, Dr. Veksler, Dr. Huang, Dr. Yu and Dr. Mercer for serving as my committee members. I also want to thank you for your brilliant comments and suggestions. I would especially like to thank everyone at Arcane Inc. for their valuable suggestions and supports.

A special thanks to my precious parents. Words cannot express how grateful I am to them for all of the sacrifices that youve made on my behalf. At the end I would like express appreciation to my beloved wife, Nobar, who spent sleepless nights with and was always my support in the moments when there was no one to answer my queries. I love you all dearly.

Abstract

Recommendation systems have been the most emerging technology in the last decade as one of the key parts in e-commerce ecosystem. Businesses offer a wide variety of items and contents through different channels such as Internet, Smart TVs, Digital Screens, etc. The number of these items sometimes goes over millions for some businesses. Therefore, users can have trouble finding the products that they are looking for. Recommendation systems address this problem by providing powerful methods which enable users to filter through large information and product space based on their preferences. Moreover, users have different preferences. Thus, businesses can employ recommendation systems to target more audiences by addressing them with personalized content. Recent studies show a significant improvement of revenue and conversion rate for recommendation system adopters.

Accuracy, scalability, comprehensibility, and data sparsity are main challenges in recommendation systems. Businesses need practical and scalable recommendation models which accurately personalize millions of items for millions of users in real-time. They also prefer comprehensible recommendations to understand how these models target their users. However, data sparsity and lack of enough data about items, users and their interests prevent personalization models to generate accurate recommendations.

In Chapter 1, we first describe basic definitions in recommendation systems. We then shortly review our contributions and their importance in this thesis. Then in Chapter 2, we review the major solutions in this context. Traditional recommendation system methods usually make a rating matrix based on the observed ratings of users on items. This rating matrix is then employed in different data mining techniques to predict the unknown rating values based on the known values.

In a novel solution, in Chapter 3, we capture the mean interest of the cluster of users on the cluster of items in a cluster-level rating matrix. We first cluster users and items separately based on the known ratings. In a new matrix, we then present the interest of each user clusters on each item clusters by averaging the ratings of users inside each user cluster on the items belonging

to each item cluster. Then, we apply the matrix factorization method on this coarse matrix to predict the future cluster-level interests. Our final rating prediction includes an aggregation of the traditional user-item rating predictions and our cluster-level rating predictions.

Generating personalized recommendation for cold-start users, or users with only few feedback, is a big challenge in recommendation systems. Employing any available information from these users in other domains is crucial to improve their recommendation accuracy. Thus, in Chapter 4, we extend our proposed clustering-based recommendation model by including the auxiliary feedback in other domains. In a new cluster-level rating matrix, we capture the cluster-level interests between the domains to reduce the sparsity of the known ratings. By factorizing this cross-domain rating matrix, we effectively utilize data from auxiliary domains to achieve better recommendations in the target domain, especially for cold-start users.

In Chapter 5, we apply our proposed clustering-based recommendation system to *Morphio* platform used in a local digital marketing agency called Arcane inc. *Morphio* is an smart adaptive web platform, which is designed to help Arcane to produce smart contents and target more audiences. In Morphio, agencies can define multiple versions of content including texts, images, colors, and so on for their web pages. A personalization module then matches a version of content to each user using their profiles. Our ongoing real time experiment shows a significant improvement of user conversion employing our proposed clustering-based personalization.

Finally, in Chapter 6, we present a summary and conclusions for this thesis. Parts of this thesis were submitted or published in peer-review journal and conferences including ACM Transactions on Knowledge Discovery from Data and ACM Conferences on Recommender Systems.

Keywords: Personalization, recommendation systems, collaborative filtering, content marketing, data mining

Contents

List of Figures

List of Tables

List of Appendices

Chapter 1

Introduction

In this chapter we review few basic concepts in recommendation systems, and then review our contributions in this thesis.

1.1 Recommendation Systems

Recommendation systems are key parts of the information and e-commerce ecosystem [11], where businesses offer an enormous number of items through different channels such as World Wide Web (WWW), Smart TVs, Digital Screens, etc. Therefore, users can have trouble finding the products that they are looking for. Recommendation systems address this problem by providing powerful methods which enable users to filter through large information and product space based on their preferences. Here are few real world examples of employing recommendation systems in e-commerce:

- *Google news* employs recommendation system to suggest news articles to its users.
- *Netflix* uses its users' feedback to recommend them movies that they would like.

In recommendation systems, there are at least two classes of entities; Users and items, where users have preferences for certain items [41]. The data itself is commonly represented as a rating matrix or a utility matrix R to show the degree of preference of users on a subset of items:

$$R = \begin{array}{c} \\ u_1 \\ u_2 \\ \vdots \\ u_n \end{array} \begin{array}{c} \begin{array}{cccc} i_1 & i_2 & \dots & i_m \end{array} \\ \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1m} \\ r_{21} & r_{22} & \dots & r_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n1} & r_{n2} & \dots & r_{nm} \end{pmatrix} \end{array}$$

where u_n represents the nth user, i_m represents the mth item, n is the number of users, m is the number of items, and r_{ij} represents the rating of user i on item j.

We assume that the values are from an ordered set. For instance, integers 1 to 5 represent the number of stars that the user gave as a rating to a movie in Netflix. In an usual recommendation scenario, matrix R is sparse. That means we only have few known ratings from each users and the rest of the ratings are unknown. The goal of recommendation systems is to predict these unknown values and prepare a list of relevant items for each user. Here are main challenges in recommendations systems:

- **Accuracy:** Businesses employ recommendations to first help users by finding their related items, and second, to increase their own revenue. Accuracy of recommendations plays a major role to achieve both these goals.

- **Scalability:** Recommendation systems need to handle millions of users and millions of items in many cases. Thus, scalability is one of the most challenging issues in recommendation systems.

- **Cold-start users:** They are users with no ratings or only few ratings. Thus, it is hard to model the interests of these users. Generating good recommendations for cold-start users is another challenge in recommendation systems.

- **Imbalanced dataset:** Number of ratings per items also usually has a power law distribution in practice. Thus, we have many ratings for few items but no ratings or very few ones for most of the items.

- **Comprehensibility:** Businesses are seeking for accurate and scalable recommendations. Yet, they tend to understand how these models generate these recommendations and target their users. This is one of the reasons that models based on K-Nearest Neighbor are popular in practice. For instance, Netflix recommends you new movies based on the movies that you have watched before.

A recommendation task can be performed in two ways; First, we may ignore the individual preferences and consider the overall preferences only. For instance, we may find a list of popular items to recommend to all user. These non-personalized recommendations are easy to achieve but less accurate considering the diverse preferences of users. Although, they result in good recommendations for cold-start users. Another approach is to consider the individual preferences to personalize the recommendations, called personalization.

1.2 Personalization

Personalization has been one the most emerging technology in the last decade [11, 22, 25, 40]. Diverse preferences of audiences forces content providers to widely employ personalization technologies. Personalization involves accommodating between individuals by finding their preferences, and employing these found preferences to locate the most relevant contents to each individual.

Personalisation techniques can be categorized into content filtering, collaborative filtering, and hybrid solutions which are a combination of the first two techniques. Generally speaking, content filtering systems focus on item properties and user profiles to determine the similarities between users and items. On the other hand, collaborative filtering systems focus on the ratings only. Thus, in collaborative filtering two items are similar if they have been rated by the same users [41].

1.3 Content Filtering

Content filtering is mainly based on item properties and user profiles. For instance, in Movie domain we may have the following profiles for each movie: actors, director, the year in which the movie is made, etc. Methods based on content filtering employ these profiles to find item-item and/or user-user similarities. These similarity is then employed for generating the recommendations. In other words, if user u is interested in item i, with a high chance she will like the items with same contents as item i. In addition, she might be interested in the items that the users with same profile as her, like.

Employing these profiles to generate the recommendations is useful. However, these profiles are usually unavailable or costly to obtain in practice. This is the main reason that methods based on collaborative filtering are more popular in both academia and industry.

1.4 Collaborative Filtering

The fundamental assumption behind collaborative filtering is if users agree about the relevance of some items, then they will likely agree about other items. For instance, if a group of users likes the same things as Mary, then Mary is likely to like the things they like which she has not seen yet [11].

1.5 Hybrid Techniques

Hybrid recommendation techniques integrate content and collaborative information to achieve higher recommendation accuracy. This content information contains user profiles, item profiles, and context information such as weather condition, location, etc. Several methods have been proposed for hybrid recommendation. KNN is an obvious choice for including this content information to improve the similarity function and consequently the recommendation accuracy. Hybrid models are represented in a multi-dimensional matrix $R : Users \times Items \times$

Content $\rightarrow \mathbb{R}$. They have been well studied in the last decade. For instance, Rendle et al. in [44] propose a Tensor Factorization technique to factorize cubic rating matrix users-items-contents. Wetzker et al. in [53] propose a hybrid solution by employing PLSA on a merged representation of user-item-tag observations. Adomavicius et al. in [2] employ ratings aggregation to reduce the multi-dimensional (contents-users-items dimensions) rating matrix to the traditional 2 dimensional rating matrix. Hariri et al. in [16] propose a KNN technique and employs inferred topics (context) to calculate the item-item similarity.

1.6 Cross-Domain Recommendations

Traditional recommendation systems assume that items belong to a single domain. However, at the present time, users rate items or provide feedback in different domains such as movies in Netflix and books in Amazon. They also express their interests in different social networks such as Facebook and Twitter. Thus, businesses intend to empower their business intelligence by incorporating cross-domain information to generate better recommendations and consequently improve their revenue. An obvious way to include the cross-domain information into our single domain scenario, is to merge domains an treat them as a single domain. This approach is also called as **collective-domain**. However, usually there are only few shared users and items between domains. Data distribution, biases, and sparsity also can be different from a domain to another. Thus, researchers propose to model each domain individually and then transfer the knowledge across domains to improve the recommendation accuracy [8]. Domains can be distinguished for the following reasons [12]:

- They may have different types of items such as movies and books.
- Different types of users can distinguish the domains such as pay-per-view users versus subscribed users.
- Finally, different context may result in different domains. For instance, user locations

and culture can separate the domains.

In general, cross-domain recommendations include a **target domain** and one or several **auxiliary domain(s)**. The goal of cross-domain recommendations is to employ these auxiliary domains to:

- Improve the recommendation accuracy in the target domain.
- Address the cold-start problem.
- Improve accuracy of recommendation for all users.
- Increase the novelty of recommendations.

Auxiliary domains can be categorized according to their users and items overlap into full overlap, users overlap, items overlap, or no overlap [8]. While domains with users overlap has attracted major studies in cross-domain recommendations.

1.7 Evaluation Metrics

Finding an offline evaluation metric that can evaluate recommendation methods comprehensively has been a subject for debate in the last few years [9, 49]. Several evaluation metrics have been proposed to address this issue. **Root Mean Squared Error (RMSE)** is perhaps the most popular metric that have been used in evaluating accuracy of predicted ratings. For a given testing set T, the recommendation system generates predicted ratings $\hat{r}_{ui}$ for testing cases $< u, i >$ where the true ratings r_{ui} are known. The RMSE between the predicted and actual ratings are then computed as follows:

$$RMSE = \sqrt{\frac{1}{|T|} . \sum_{<u,i>\in T} (r_{ui} - \hat{r}_{ui})^2} \tag{1.1}$$

Mean Absolute Error (MAE) is also an alternative evaluation metric, given by:

$$MAE = \sqrt{\frac{1}{|T|}. \sum_{<u,i>\in T} |r_{ui} - \hat{r}_{ui}|} \tag{1.2}$$

Compared to MAE, RMSE excessively penalizes large errors [47]. As mentioned earlier, both RMSE and MAE evaluate rating prediction accuracy. However, in many real world application of recommendation system we only want to find the few top recommendations for each user. In other words, we may do not care how bad our rating prediction is as long as our system ranks the top 10 or 20 items for each user precisely. **Top-N recommendation task** is designed as a ranking based evaluation metric [24]. This evaluation metric is proposed by Koren et al. in [23]. Assume T as the set of all ratings in the test set with the highest rank ($r_{ui} = 5$ when $r_{ui} \in [1, 5]$). For each test example $< u, i >$ in T, 1000 items are randomly selected from the set of items. They then predict the preference of user u on those selected items plus item i. They form a ranked list by ordering all the 1001 items according to their predicted ranking values in matrix R. Finally, They form a top-N recommendation list by picking the top N ranked items from the list. If they have item i in the top-N list they have a hit (the test item i is correctly recommended to the user). Otherwise they have a miss. Chances of a hit obviously increase with N. They measure the recall based on the number of hits in a list of N recommendations as follows:

$$Recall(N) = \frac{\#hits}{|T|}.$$

Thus, by increasing the recall we will have more interesting items for each user in her personalized top-N list.

1.8 Adaptive Web

Content Marketing is one of the fastest growing industries in World Wide Web (WWW). It is any marketing that involves the creation and presentation of media and publishing content in

WWW or other digital channels to acquire and attract audiences. Nowadays, businesses switch from static web to adaptive web, where they can make different versions of content to target more audiences with diverse preferences. This personalization task is almost different from the product personalization such as recommendation of movie or books where we have millions of products. Adaptive web deal with only few versions of contents. Having even few feedback from users is rare in this scenario as they are mostly new users(cold-start users).

The traditional personalization task in adaptive webs has been commonly done based on manually segmentation of user based on predefined rules. For instance, users who live in Canada, users who use Internet Explorer for browsing, etc. They then employ A/B testing of different contents on different segment of users to find the best matches. A new user then will be mapped to one of these predefined segments and will be presented by this segment's matched version. However, this traditional supervised method is costly and the predefined rules can only cover few users.

1.9 Contributions of this Thesis

In this thesis, we have several novel contributions to improve the accuracy, scalability, and comprehensibility of current popular solutions in recommendation systems, and especially in recommendation systems with high data sparsity. As mentioned in Section 1.1, recommendation systems apply different data mining techniques on the known ratings of users on items to predict the unknown ratings. As a novel contribution, we turn the direct user-item interests into a higher level. We first employ the traditional rating matrix to explore the similarity between the users and items and cluster them separately into multiple user and item clusters. Then, we capture the averaged interests of users in each user clusters on the items in the item clusters in a new cluster-level rating matrix. Thus, in this new rating matrix we generalize the known interests. For instance, the fact that John is interested in a song in the Rock genre can be generalized into an assumption that John and his similar users are interested in other songs

in the Rock genre as well. We can now apply every proposed recommendation techniques for the traditional rating matrix on this new cluster-level rating matrix to compute the unknown cluster-level ratings. Eventually, we aggregate these two level of rating predictions as our final ratings. Thus, our cluster-level rating matrix works as an wrapper and can be employed in any recommendation systems. In Chapter 3, we employ our proposed cluster-level rating matrix to improve the accuracy of few well-studied recommendation models.

As mentioned earlier, collaborative filtering methods employ known ratings to find users and items similarities and also model users' preferences. However, for cold-start users there are not sufficient known ratings to be employed for the personalization task. Thus, generating personalized recommendations for cold-start users is a big challenge in recommendation systems. Employing any available information from these users is crucial to improve their recommendation accuracy. For instance, available interests of users in a Book domain can be exploited to generate better recommendations for them in a Movie domain. In Chapter 4, we extend our cluster-level rating matrix from single-domains into cross-domains. Thus, in our extended clustering-based recommendation system we generalize the users' interests in the auxiliary domains to improve recommendation accuracy in the target domain.

Finally in Chapter 5, we apply our proposed clustering-based recommendation system to *Morphio* platform used in a local digital marketing agency, called Arcane inc. *Morphio* is an smart adaptive web platform, which is designed to help Arcane to produce smart contents and target more audiences. In Morphio, web developers can define multiple versions of content for their web pages. Our proposed clustering-based personalization tool then shows each user of them a personalized version. We first cluster old users using their profiles and page visits. We then find the best version for each of these generated clusters employing an A/B testing. A trained deep learning model then soft assign new users into these clusters, which eventually results in their matched versions of content. In addition, we also employ our proposed clustering-based recommendation system to suggest smart contents for current web pages. We first analyse the impact of each piece of content in these web pages. In a new matrix, we then

present these generated impact values between pieces of content and pages. Eventually, we apply our proposed clustering-based recommendation system to predict the unknown values in this new matrix. Hence, this module allows agencies to improve their produced content using our data-driven suggestions and insights.

Chapter 2

Literature Review

2.0.1 K-Nearest Neighbor (KNN)

Collaborative filtering based K-Nearest Neighbor also known as user-user collaborative filtering was the first of the automated CF methods [11]. KNN based recommendation models are popular in practice as they do not need an offline training. They use a similarity function $s : U \times U \rightarrow \mathbb{R}$ to compute a neighborhood $N \subseteq U$ of similar users to user u, where U is the set of all users. Once we compute neighborhood N, we can predict preference of user u on item i as follows:

$$r_{u,i} = \bar{r}_u + \frac{\sum_{u' \in N} s(u, u')(r_{u',i} - \bar{r}_{u'})}{\sum_{u' \in N} s(u, u')} \tag{2.1}$$

where $\bar{r}_u$ is the mean rating of user u. Some users tend to give higher ratings than others. That is the reason we subtract their ratings by their mean rating to eliminate the possible bias. In Equation 2.1, a question is how many neighbors to select. In some systems, all users are selected as the neighbors. However, computing all user-user similarities is complex for millions of users and items. Thus, the large number of neighbors reduces the scalability of KNN.

The same approach can be employed to compute the item-item similarity and a item neighborhood. Thus, by employing similarity function $s : I \times I \rightarrow \mathbb{R}$ to compute a neighborhood

$N \subseteq I$, we can predict predict rating $r_{u,i}$ as follows:

$$r_{u,i} = \bar{r}_i + \frac{\sum_{i' \in N} s(i,i')(r_{u,i'} - \bar{r}_{i'})}{\sum_{i' \in N} s(i,i')} \tag{2.2}$$

where I is set of all items and $\bar{r}_i$ is the mean rating of item i. Cosine similarity, Spearman rank correlation, and Pearson correlation are well-known similarity functions which are widely used in methods based on KNN [11, 45].

2.0.2 Matrix Factorization

Compared to KNN methods, methods based on factorization results in better recommendation accuracy with less complexity. However, they need to be trained offline, and need to be re-trained for new users and items. Matrix Factorization (MF) decomposes the ratings matrix, R, into two lower dimension matrices Q and P where:

$$R = P \times Q^T \tag{2.3}$$

Q and P contain corresponding latent vectors of each user and each item(Figure 3.1.a). Let's assume $l \ll m, n$ is the length of the latent vectors in these two matrices. Thus for each user u we have the following latent vector p_u:

$$p_u = [p_{1u}, p_{2u}, \ldots, p_{lu}],$$

and for each item i we have the following latent vector q_i:

$$q_i = [q_{1i}, q_{2i}, \ldots, q_{li}].$$

Matrix Factorization is based on the singular value decomposition (SVD) technique for finding latent vectors in information retrieval [26].

To find proper P and Q, a training algorithm can be started by a random initialization of

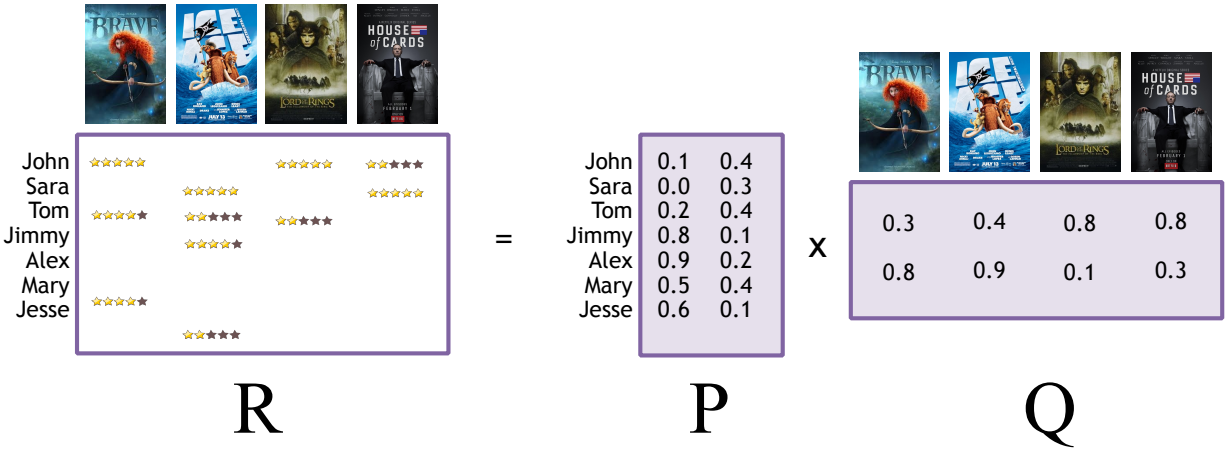


Figure 2.1: Factorizing rating matrix R into lower dimension matrices P, and Q, where $R = P.Q^T$.

these matrices. In every learning step, it then tries to change the initialized variables in a way that $P.Q^T$ converges to the known values of R. In the prediction case, the product of learned matrices will be used to predict the unknown $r_{ui}s$ as follows:

$$r_{ui} = q_i^T p_u \tag{2.4}$$

MF characterizes every user and item by corresponding them a latent vector. These latent vectors can be considered as the hidden profiles for users and items. Also, a bias value is typically corresponded to each user and each item to reflect their mean ratings. Adding these biases, the above statement will change to:

$$r_{ui} = q_i^T p_u + b_{ui}, \tag{2.5}$$

where

$$b_{ui} = b_u + b_i.$$

This method is called **Biased Matrix Factorization (BMF)**. Let's define the error as the actual rating minus the predicted value in each step, e_{ui}. In a training task, we should find

appropriate values for P and Q which minimization the following objective function:

$$min(\sum_{(u,i)\in R}(r_{ui} - q_i^T p_u - b_{ui})^2 + \lambda_1.(|q_i|^2 + |p_u|^2) + \lambda_2.(|b_u|^2 + |b_i|^2)) \quad (2.6)$$

Regularization values λ_1 and λ_2 prevent over-fitting on the model and keep the latent values small. P and Q can be learned using several proposed methods such as the stochastic gradient descent technique [26]. Simon Funk[1] [26] popularized a stochastic gradient descent optimization of Equation 2.6 wherein the algorithm loops through all ratings in the training set. For each given known rating r_{ui}, the system predicts $\hat{r}_{ui}$ and computes the associated prediction error e_{ui} as follows:

$$e_{ui} = r_{ui} - \hat{r}_{ui}$$

Then it modifies the parameters step wise in the opposite direction of the gradient in several iterations(Algorithm 1).

Algorithm 1 The stochastic gradient descent optimization for Objective function 2.6.

for $count = 1, ..., \#Iterations$ **do**
  for $r_{ui} \in R$ **do**
    $e_{ui} \leftarrow r_{ui} - \hat{r}_{ui}$
    $q_i \leftarrow q_i + \gamma(e_{ui}.p_u - \lambda_1.q_i)$
    $p_u \leftarrow p_u + \gamma(e_{ui}.q_i - \lambda_1.p_u)$
    $b_u \leftarrow b_u + \gamma(e_{ui} - \lambda_2.b_u)$
    $b_i \leftarrow b_i + \gamma(e_{ui} - \lambda_2.b_i)$
  end for
end for

This popular approach is easy to implement and fast in running time [26]. In practice, new users, items, and ratings are regularly added into a recommendation system. Thus, matrix factorization models need to be trained frequently. Yet, scalability and high accuracy of these models have made them extremely popular in recommendation systems.

[1] http://sifter.org/ simon/ journal/20061211.html

2.0.3 Functional Matrix Factorizations

Zhou et al. in [57] propose a functional matrix factorization to improve the recommendation accuracy for cold-start users. This functional matrix factorization employs a decision tree and an initial interview process to produce a profile for the new users and adopt this profile to generate more accurate recommendations for these users. The decision tree of the initial interview contains several nodes as interview questions, enabling the recommendation system to query a user adaptively according to her prior responses. Assume a_u includes users u answers to a designed interview. To factorize rating matrix R into latent matrices P and Q, they define a decision tree function T where $p_u = T(a_u)$. Thus, function $T(a_u)$ maps users u to an appropriate latent vector p_u. Employing this generated p_u, this functional matrix factorization predict the unknown ratings for new user u.

2.0.4 Neighborhood-Aware Models

As discussed in Section 2.0.1, KNN models employ the neighborhood information to generate recommendations. While factorization models result in a higher accuracy of recommendations. Several methods have been proposed in which neighborhood information is employed beside matrix factorization to improve the recommendation accuracy even further [23, 24, 26, 52]. For instance, Töscher et al. [52] present a neighbourhood-aware matrix factorization in which they include neighbourhood information into the traditional matrix factorization. Their proposed algorithm computes three level of predictions for every user-item pair: a traditional rating prediction r_{ui} based on matrix factorization(similar to Section 2.0.2); a rating prediction r_{ui}^{user} based on user neighbourhoods; and finally a rating prediction r_{ui}^{item}, which is based on item neighbourhoods. A combination of these three rating predictions will generate the final recommendations. The rating prediction r_{ui}^{user} is computed as follows:

$$r_{ui}^{user} = \frac{\sum_{v \in U_J(u)} c_{uv}^{user} r_{vi}}{\sum_{v \in U_J(u)} c_{uv}^{user}}$$

where $U_J(u)$ denotes the set of J users with highest correlation to user u. These correlations are reached by counting the number of co-rating of users. And r_{vi} is the predicted rating of user v on item i which will be calculated similar to r_{ui}. r_{ui}^{item} is also calculated as follows:

$$r_{ui}^{item} = \frac{\sum_{j \in U_J(i)} c_{ij}^{item} r_{uj}}{\sum_{j \in U_J(i)} c_{ij}^{item}}$$

where $U_J(i)$ denotes the set of J items with highest correlation to item i. It is mentioned in [52] that including this neighbourhood information improves MF's recommendation accuracy. However, its complexity is still sensitive to the choice of J.

2.0.5 Clustering-Based Recommendations

Applying clustering on the rating matrix R is also another approach for generating the recommendations [3,7,13–15,20,54]. In 1999, O'Connor et al. proposed a very first use of clustering algorithm in recommendation systems [7]. They apply clustering on rating matrix R to cluster items. For each cluster, they then compute rating predictions for test set based on KNN. Their results show a high improvement of scalability for KNN model but they had mixed results of improvements for recommendation accuracy. George et al. in [14] propose an efficient recommendation model based on co-clustering of users and items. They consider user cluster rating averages and item cluster rating averages beside user mean ratings and item mean ratings in the rating prediction function. In other words, to predict r_{ui} they employ mean rating of user u and mean rating of item i beside the averaged ratings of items inside the item cluster that i belongs to and the averaged ratings of users inside the user cluster that user u belongs to. Obviously, their proposed model achieve a high scalability but not a higher accuracy comparing to MF based models.

Gueye et al. in [15] integrate cluster mean ratings into matrix factorization to improve the traditional matrix factorization models. They first employ clustering to partition items and users into several clusters. They then use the following rating prediction function to generate

the recommendations:

$$r_{ui} = p_u.q_i^T + \mu_{C_i} + b_{u,C_i} + b_i$$

where C_i is the cluster that item i belongs to, b_i is the mean rating for item i; b_{u,C_i} is the mean rating of user u on the items inside cluster C_i and is calculated as follows:

$$b_{u,C_i} = \frac{1}{|C_i|} \sum_{j \in C_i} (r_{uj} - \mu_{C_i}).$$

In addition, Xu et al. [54] employ clustering in a different way to improve the accuracy of recommendation. They cluster items and users into subgroups where each user or item can belong to more than one cluster. Their main idea is to apply number of collaborative filtering algorithm in each subgroup and then merge the prediction results together. They consider clusters reasonably large to possess enough known ratings in each subgroup.

Xue et al. in [56] also propose a KNN method adopting clustering. Methods based on KNN for collaborative filtering determine the similarity between two users by comparing their ratings on a set of items. As mentioned earlier, KNN approaches have been shown to suffer from two fundamental problems: data sparsity and difficulty in scalability. Xue et al. cluster users from the training data to provide the basis for data smoothing and neighborhood selection. For each given user u, they first find the most similar clusters to user u and then employ the users inside the selected cluster to generate the recommendations in a KNN process. As a result, they provide higher accuracy as well as increased efficiency in recommendations [56].

2.0.6 Implicit vs. Explicit Feedback

Marlin et al. in [31] experimentally show that the ratings data in usual recommendation systems does not have a balanced distribution and they are missing not in random. Users are free to choose which items to rate [49]. Thus, unobserved ratings are likely low ratings. In other words, users watch movies that they think they may like. Hence, the rated movies can be

Figure 2.2: Users watch movies that they think they may like. Hence, a rated movie can be considered as an interesting movie for a user.

considered as positive feedback by users no matter what the ratings are. In a new utility matrix $R' = [r'_{ui}]$, we may represent rating matrix R as follow:

$$r'_{ui} = \begin{cases} 1 & \textit{if } r_{ui} \textit{ is observed} \\ 0 & \textit{otherwise} \end{cases} . \tag{2.7}$$

Figure 2.2 illustrates an example scenario for rating matrix R and R'. Employing this implicit feedback has been shown significantly helpful in improving the recommendation accuracy [9, 23, 49, 50].

SVD++ [23, 25] is one of the earliest models that includes implicit feedback into matrix factorization model to improve its accuracy. Let's assume $N(u)$ contains implicit feedback or all the items that user u have rated. SVD++ corresponds each item i with two latent vectors $q_i, y_i \in \mathbb{R}^l$ and employs the following rating prediction function:

$$r_{ui} = q_i^T . p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} q_i^T . y_j + b_{ui} \tag{2.8}$$

or simply:

$$r_{ui} = q_i^T (p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j) + b_{ui} \tag{2.9}$$

where user u is modeled with its corresponding latent vector, p_u, plus the items that she has rated, $|N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j$. Including the implicit feedback using this factorization technique gives a high scalability to this model. SVD++ also has shown a promising improvement of prediction accuracy in practice [26]. Note that $\sum_{j \in N(u)} q_i^T . y_j$ represents the implicit feedback as there is not any rating information on it. In other words, they assume each rating as a positive feedback no matter what the rating is. Koren et al. in [23] extend their own SVD++ model by including the explicit neighborhood information beside the implicit one as follows:

$$r_{ui} = q_i^T \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} (r_{uj} - b_{uj}) x_j \right) + b_{ui} \tag{2.10}$$

where latent vector x_j represents the explicit neighborhood relation for item j, and $|N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} (r_{uj} - b_{uj}) x_j$ represents explicit neighborhood information of user u. They call this model as **Asymmetric SVD (Asysvd)**. Asysvd shows a very high quality of rating prediction and even outperforms SVD++ in practice [23, 24]. Koren employs the stochastic gradient descent technique in both SVD++ and Asysvd to learn the corresponded parameters [24]. Algorithm 2 shows the proposed training algorithm for Asysvd model in [24].

There are also several other works those try to factorize the implicit rating matrix R' directly. For instance, Cremonesi in [9] employs pure SVD to factorize matrix R' as follows:

$$R' = P.\Sigma.Q^T$$

where P is a $n \times l$ orthonormal matrix, Q is a $m \times l$ orthonormal matrix, and Σ is a $l \times l$ diagonal matrix containing the first l singular values. Note that rating matrix R is an sparse matrix so we do not have many observed ratings in the training time and factorization can be done quickly. However, all the values in matrix R' are known values (0 or 1), thus, factorizing a full-filled matrix with millions of rows and column is computationally expensive. To address this issue, Steck in [49] proposes a new **Alternative Least Squares (ALS)** based learning model to include implicit information from matrix R' more efficiently. He assumes different weights for observed and unobserved ratings as follows:

Algorithm 2 A training algorithm for Asysvd model based on stochastic gradient descent technique.

for $count = 1, ..., \#Iterations$ **do**
  for $u = 1, ..., n$ **do**
    %Computes the components independent of i
    $p_u \leftarrow p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} (r_{uj} - b_{uj}) x_j$
    $sum \leftarrow 0$
    for $i \in N(u)$ **do**
      $r_{ui} = p_u . q_i^T + b_{ui}$
      $e_{ui} = r_{ui} - r_{ui}$
      %Accumulate information for gradient steps on x_i and y_i
      $sum \leftarrow sum + e_{ui} . q_i$
      % Perform gradient step on q_i, b_i, b_u:
      $q_i \leftarrow q_i + \gamma(e_{ui} . p_u - \lambda . q_i)$
      $b_u \leftarrow b_u + \gamma(e_{ui} - \lambda . b_u)$
      $b_i \leftarrow b_i + \gamma(e_{ui} - \lambda . b_i)$
    end for
    for $i \in N(u)$ **do**
      %Perform gradient step on x_i, y_i
      $x_i \leftarrow x_i + \gamma |N(u)|^{-\frac{1}{2}} . (r_{ui} - b_{ui}) . sum - \lambda x_i$
      $y_i \leftarrow y_i + \gamma |N(u)|^{-\frac{1}{2}} . sum - \lambda y_i$
    end for
  end for
end for

$$W_{ui} = \begin{cases} w_{obs} & \text{if } r_{ui} \text{ is observed} \\ w_m & otherwise(w_m < w_{obs}) \end{cases}. \tag{2.11}$$

To factorize rating matrix $R = r_m + P.Q^T$, he then employs weighting function W_{ui} in the following objective function:

$$\sum_{\text{all } u} \sum_{\text{all } i} W_{ui}. \left\{ (r_{ui} - \hat{r}_{ui})^2 + \lambda.(\sum_{j=1:l} P_{i,j}^2 + Q_{u,j}^2) \right\}, \tag{2.12}$$

where r_{ui} can be an observed or non-observed rating; $\hat{r}_{ui} = r_m + p_u.q_i^T$ is the predicted rating. W_{ui}, and λ are assumed fixed tuning parameters, which optimized via a validation set as to maximize recommendation accuracy. ALS is then applied to find a (close to) minimum solution of Equation 2.12 by employing gradient descent technique. At each step, one of two matrices P and Q is assumed fixed, which turns the updating process of the other matrix into a quadratic optimization problem that can be solve exactly through equating the gradient of Equation 2.12 to zero. This results in the following updating equation for each corresponded latent vector Q_i (P is assumed fixed):

$$Q_i = (R_i - r_m)\hat{W}^{(i)}P(P^T\hat{W}^{(i)}P + \lambda.tr(\hat{W}^{(i)})I)^{-1}, \tag{2.13}$$

where vector R_i includes all the ratings on item i from rating matrix R; $\hat{W}^{(i)} \in \mathbb{R}^{n \times n}$ is the diagonal matrix containing the i^{th} row of matrix W; $I \in \mathbb{R}^{l \times l}$ is the identity matrix.

In the next step, Q will be assumed as a fixed value, which turns the updating equation for each P_u as follows:

$$P_u = (R_u - r_m)\hat{W}^{(u)}Q(Q^T\hat{W}^{(u)}Q + \lambda.tr(\hat{W}^{(u)})I)^{-1}, \tag{2.14}$$

where $\hat{W}^{(u)} \in \mathbb{R}^{m \times m}$ is the diagonal matrix containing the u^{th} row of matrix W. Note that for unobserved ratings we have $R_{i,u} - r_m = 0$. Thus, updating Equations 2.13 and Equation 2.14 can

be rewritten simpler and more computationally efficient, which is thoroughly discussed in [49]. Parallelization ability, and simple updating process for new-coming users are two advantages of this updating mechanism.

Chapter 3

Leveraging Clustering to Improve Collaborative Filtering

3.1 Introduction

As mentioned in Section 2.0.2, extensive work on Matrix Factorization (MF) has been done recently as it provides very promising collaborative filtering solutions for recommendation systems. Additional extensions, such as neighbor-aware models (Section 2.0.4), have been shown to improve these results further. Recommendation methods based on MF show a good prediction accuracy and scalability. Yet, employing clustering on the set of users and items has been a basic and practical solution in recommendation systems (Section 2.0.5). In this chapter, we integrate the advantages of both disciplines to achieve a higher recommendation accuracy.

We first cluster users and items separately into multiple user clusters and items clusters. Because of the large number of users and items, we employ MF to generate corresponding latent vectors for users and items in a lower dimension. We then apply K-Means on these latent vectors to clusters items and users. We then capture the common interests between the cluster of users and the cluster of items in a cluster-level rating matrix. We make a "coarse" matrix where each cluster of users(items) is considered as one user(item) entity, and the ratings represent the

averaged ratings of the users inside the user clusters on the items inside the item clusters. By applying MF or any other collaborative filtering methods on this generated coarse matrix, we can produce cluster-level rating predictions for unknown ratings. Finally, we aggregate these two levels of predictions to improve the recommendation accuracy further.

Extensive experimental results show that our new approach, when applied to a variety of existing collaborative filtering based methods, including Biased Matrix Factorization (BMF) and Asymmetric SVD (Asysvd) in Sections 2.0.2 and 2.0.6, improves their rating prediction accuracy. We also evaluate how the quality and quantity of these clusters impact these improvements.

As discussed in Section 2.0.5, employing clustering to categorize collaborative information, and using these clusters to predict the unknown preferences, has been employed before in a number of works such as [54], [14], [20], and [15]. However, many of these methods mainly focus on the individual clusters and ignore the shared interest between the clusters. Or, they proposed expensive probabilistic models which cannot be easily integrated with the-state-of-the-art collaborative filtering methods. Moreover, our extension approach is easy to implement and can be applied as a "wrapper" on any other collaborative filtering methods.

This chapter is structured as follows. Section 3.2 describes our proposed model to extend current factorized methods in collaborative filtering. In Section 3.3 we describe our empirical experimental results. In Section 3.5 we review previous works related to factorized collaborative filtering methods and recommendation models based on clustering.

3.2 The Proposed Models

In a general CF problem, we have a set of users $U = \{u_1, u_2, \ldots, u_n\}$ and a set of items $I = \{i_1, i_2, \ldots, i_m\}$ that are accompanied by a rating matrix $R = [r_{ui}]_{n \times m}$ where r_{ui} represents the rating of user u on item i. Collaborative filtering consists of predicting unknown r_{ij}s based on the known $r_{i'j'}$ s inside the rating matrix R and similar neighbours.

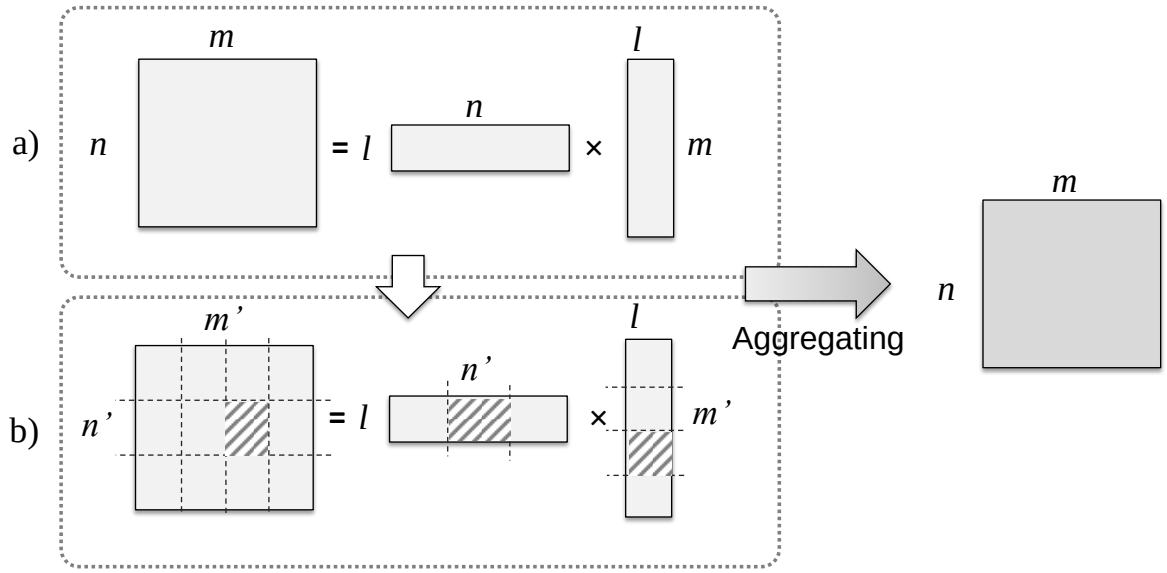


Figure 3.1: Factorizing rating matrix R into latent matrices P and Q (a) and clustering these found latent matrices to produce cluster-level rating matrix R_C. (b) Factorizing rating matrix R_C and aggregating these two levels of latent vectors to generate the recommendations.

Employing clustering is a classic approach in recommendation system for dividing the set of users and items into different categories and making similarity-based recommendations for each of these categories (Section 2.0.5). However, the traditional methods mainly focus on the individual clusters and ignore the shared interest between the clusters. Using the shared preferences of the categories has three advantages opposed to the common neighbourhood models:

1. It generalizes the preference of users on items regarding the possible categories that they belong to. For example, user u may belong to the category 'Adults' and item i may belong to the category 'Cartoons'. Thus, in addition to considering if user u is interested in item i, it deliberates if the category 'Adults' shares any preferences with the category 'Cartoons' in general.

2. It considers deeper similarities. Item-item models check if user u is interested in item i and its similar items. User-user models also check if user u and its similar users are interested in item i. In our approach, we check to see if user u and its similar users are

interested in item i and its similar items on average.

3. The clusters can be interpreted using the content information and employed to justify the recommendation. Thus, using this information, the recommendation model is much more comprehensible. For instance, knowing that a group of users likes 'Cartoons' and not 'Dramas' and/or 'Classic Movies' results in a higher comprehensibility.

Alex Beutel et al. in [3] also describe many other advantages of employing clustering in a recommendation model. To cluster users and items, we first apply the biased matrix factorization on the known ratings to learn the latent vectors of each user and item (Figure 3.1.a). K-means is then applied to these latent vectors with different selection of K (number of clusters) to find possible categories of items and users (Figure 3.1.b). Employing latent vectors to cluster sparse data has been successfully used in the literature, such as [55]. As mentioned in Section 2.0.6, rating matrices are typically sparse in recommendation systems. Hence, using latent vectors helps to reduce the complexity of clustering these large and sparse datasets because 1) there is no sparsity in these latent vectors, 2) they are in a much lower dimension.

We consider the common preferences between these clusters in a "coarse" matrix R_C. In this new rating matrix, every r_{C_u,C_i} represents the mean rating of the users inside the category C_u on the items inside the category C_i, as follows:

$$R_C = \begin{array}{c} \\ C_{u_1} \\ C_{u_2} \\ \vdots \\ C_{u_{n'}} \end{array} \begin{array}{c} \begin{array}{cccc} C_{i_1} & C_{i_2} & \dots & C_{i_{m'}} \end{array} \\ \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1m'} \\ r_{21} & r_{22} & \dots & r_{2m'} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n'1} & r_{n'2} & \dots & r_{n'm'} \end{pmatrix} \end{array}$$

where $n' < n$ and $m' < m$ are the number of clusters for users and items respectively. However, because of the small number of known ratings, r_{C_u,C_i} is not accurate enough for all pairs of

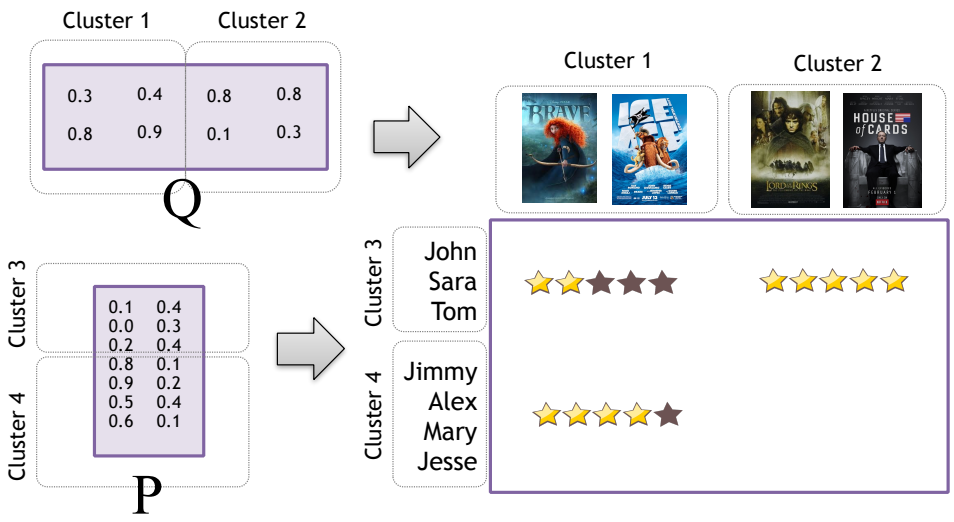


Figure 3.2: Clustering latent matrices P and Q to achieve clusters of users and items and producing the coarse matrix. The coarse matrix generalizes preferences of users into a cluster-level which leads to less sparsity in R^c.

clusters. To resolve this issue, we consider r_{C_u,C_i} as an known rating in matrix R_C, only if we have enough observed ratings between clusters C_u and C_i. We then employ MF or other collaborative filtering methods on this coarse matrix to predict all unknown r_{C_u,C_i}s. Figure 3.2 illustrates an example of applying clustering on latent matrices P and Q and generating the coarse matrix R_c.

3.2.1 Clustering-Based Matrix Factorization

As mentioned in Section 2.0.2, Biased Matrix Factorization (BMF) decomposes the ratings matrix, R, into two lower dimension matrices Q and P plus biases:

$$r_{ui} = q_i^T p_u + b_{ui} \tag{3.1}$$

where

$$b_{ui} = b_u + b_i$$

As mentioned earlier, we can apply BMF on the coarse matrix R_C to predict its unknown

ratings. By corresponding each cluster a latent vector of length l and a bias value, a cluster-level rating prediction function for user u and item i, r_{C_u,C_i}, will be as follows:

$$r_{C_u,C_i} = q_{C_i}^T p_{C_u} + b_{C_i} + b_{C_u} \tag{3.2}$$

C_i and C_u are the clusters that item i and user u belong to, and q_{C_i} and p_{C_u} are the corresponding latent vectors of these categories. Thus, instead of predicting a rating for pairs of users and items, Equation 3.2 predicts a rating for the clusters those they belong to. However, the common preferences between clusters are too general to be employed solely for the prediction purpose. Hence, we use a fusion of the traditional biased matrix factorization model and the predictor function 3.2 in a final predictor function as follows:

$$r_{ui} = T_\alpha(q_i, q_{C_i})^T T_\alpha(p_u, p_{C_u}) + T_\beta(b_u, b_{C_u}) + T_\beta(b_i, b_{C_i}) \tag{3.3}$$

where T_α is a trade-off function defined as follows:

$$T_\alpha(x, y) = (1 - \alpha).x + \alpha.y \tag{3.4}$$

$0 \leq \alpha \leq 1$ and $0 \leq \beta \leq 1$ control the effect of both models in the final predictor function. We name this fusion form Clustering-Based Matrix Factorization (CBMF) in our experimental results.

We train BMF and CBMF models in the final model simultaneously. We optimize the parameters regrading the following objective function:

$$\begin{aligned} min(&\sum_{(u,i)\in R} (r_{ui} - T_\alpha(q_i, q_{C_i})^T T_\alpha(p_u, p_{C_u}) + T_\beta(b_u, b_{C_u}) + T_\beta(b_i, b_{C_i}))^2 \\ &+ \lambda_3.(|q_i|^2 + |p_u|^2) + \lambda_2.(|b_u|^2 + |b_i|^2) + \lambda_2.(|q_{C_i}|^2 + |p_{C_u}|^2) + \lambda_4.(|b_{C_u}|^2 + |b_{C_i}|^2)) \end{aligned} \tag{3.5}$$

The parameters are determined by minimizing the associated regularized squared error function through gradient descent. Algorithm 3 presents the training process of our proposed

model.

Algorithm 3 Our proposed CBMF's updating algorithm

% Inputs: Users and items cluster assignments, training data points, and randomly initialized parameters.
% Outputs: Trained parameters including P,Q, P_C, Q_C, and the corresponded biases values.
repeat
  for all $r_{ui} \in R$ **do**
    $\hat{q}_i \leftarrow T_\alpha(q_i, q_{C_i})$
    $\hat{p}_u \leftarrow T_\alpha(p_u, p_{C_u})$
    $\hat{b}_{ui} \leftarrow T_\beta(b_{ui}, b_{C_uC_i})$
    $\hat{r}_{ui} \leftarrow \hat{b}_{ui} + \hat{q}_i^T \hat{p}_u$
    $e_{ui} \leftarrow r_{ui} - \hat{r}_{ui}$
    % Perform gradient step:
    $q_i \leftarrow q_i + \gamma_1(e_{ui}.\hat{p}_u - \lambda_1.q_i)$
    $p_u \leftarrow p_u + \gamma_1(e_{ui}.\hat{q}_i - \lambda_1.p_u)$
    $q_{C_i} \leftarrow q_{C_i} + \gamma_2(e_{ui}.\hat{p}_u - \lambda_2.q_{C_i})$
    $p_{C_u} \leftarrow p_{C_u} + \gamma_2(e_{ui}.\hat{q}_i - \lambda_2.p_{C_u})$
    $b_i \leftarrow b_i + \gamma_3(e_{ui} - \lambda_3.b_i)$
    $b_u \leftarrow b_u + \gamma_3(e_{ui} - \lambda_3.b_u)$
    $b_{C_i} \leftarrow b_{C_i} + \gamma_4(e_{ui} - \lambda_4.b_{C_i})$
    $b_{C_u} \leftarrow b_{C_u} + \gamma_4(e_{ui} - \lambda_4.b_{C_u})$
  end for
until for limited number of epochs

3.2.2 Employing More Clusters

In Section 3.2.1, we cluster both user-related latent vectors and item-related latent vectors once and employ those found clusters in our proposed CBMF model. However, clustering items and users into different sizes of clusters results in a variety of informative clusters. For instance, clustering movies into 10 clusters may capture more general similarities among items such as genre. Although, clustering the movies into 500 clusters may distinguish movies based on slight differences such as the year of production and so on. Thus, in this section we start from small number of clusters ($m' = n' = 10$ in our experiment) and gradually increase m' and n' to find the clusters in different sizes. Koren et al. in [26] shows that employing more informative parameters in the prediction model will improve rating prediction accuracy. Thus, we expect

to improve rating prediction accuracy by employing more clusters in different sizes(as they provide different level of information).

Assume $C_U^\Sigma = \{\cup_\sigma C_U^{n'_\sigma}\}$ and $C_I^\Sigma = \{\cup_\sigma C_I^{m'_\sigma}\}$ contain all found clusters for users and items in different sizes, where $C_U^{n'_\sigma}$ contains the found n'_σ clusters of users and $C_U^{m'_\sigma}$ contains the found m'_σ clusters of items. To employ all clusters inside C_U^Σ and C_I^Σ in our CBMF model, we correspond each cluster a latent vector of length l and a bias value. We define the final latent corresponding vectors $p_{C_u}^F$ ($q_{C_i}^F$) as the sum of the latent vectors of , and $b_{C_u}^F$ ($b_{C_i}^F$) as the sum of the biases of, all user (item) clusters that user u (item i) belongs to:

$$p_{C_u}^F = \sum_{C_u \in C_U^\Sigma} P_{C_u}, \tag{3.6}$$

$$q_{C_i}^F = \sum_{C_i \in C_I^\Sigma} q_{C_i}. \tag{3.7}$$

The final prediction function for this extended CBMF model is as follows:

$$r_{ui} = T_\alpha(q_i, q_{C_i}^F)^T T_\alpha(p_u, p_{C_u}^F) + T_\beta(b_u, b_{C_u}^F) + T_\beta(b_i, b_{C_i}^F) \tag{3.8}$$

In Section 3.3.4, we show that adding more number of clusters will improve the prediction accuracy of this extended CBMF model further. While adding more clusters will increases the complexity.

3.2.3 Integrating Cluster-Level Preferences With Various Methods

As mentioned in Section 3.1, our extension approach is easy to implement and can be applied in most collaborative filtering methods. There are two common approaches in neighbourhood aware matrix factorization models: 1) *item-item* models, which consider if user u is interested in item i and its similar items. 2) *user-user* models that consider if user u and its similar users are interested in item i. In the literature [52] [24], an integration of both item-item and

user-user models sometimes has been applied in the final predictor function to achieve finer accuracy. However, item-item models are usually preferable as they have less space and time complexity. This is because of the typically larger number of users in recommendation systems. As described in Section 2.0.4, neighborhood aware models employ the similarities between users and items to improve recommendation accuracy. However, they usually need to compute all pairwise similarities between items or users, and its complexity grows quadratically with the input size [24]. Koren in [23] [24] solves this limitation by factoring the neighbourhood model, which scales both item-item and user-user implementations linearly with the size of the data [24]. Thus, he effectively integrates implicit and explicit neighbourhood information to extend the Biased MF model.

In the Asymmetric SVD2.0.6, Koren proposes a factorized item-item model as follow:

$$r_{ui} = q_i^T \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) x_j \right) + b_{ui} \tag{3.9}$$

To predict the unknown cluster-level ratings in matrix R_C and integrate them with the traditional predictions in rating matrix R, we assign three latent vectors $q_{C_i}, y_{C_i}, x_{C_i} \in \mathbb{R}^l$ to each category of items, and a latent vector $p_{C_u} \in \mathbb{R}^l$ to each category of users that reflects their cluster-level implicit and explicit impact on the ratings. The new predictor function is as follow:

$$\begin{aligned} r_{ui} = T_\alpha(q_i, q_{C_i})^T \left(T_\alpha(p_u, p_{C_u}) + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} T_\alpha(y_j, y_{C_j}) + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_u - b_j) T_\alpha(x_j, x_{C_j}) \right) \\ + T_\beta(b_u, b_{C_u}) + T_\beta(b_i, b_{C_i}) \end{aligned} \tag{3.10}$$

We call this model CB-Asysvd. The parameters again are determined by minimizing the associated regularized squared error function through gradient descent. Algorithm 4 presents the training algorithm for this recommendation model.

Algorithm 4 Our proposed CB-Asysvd updating algorithm

% Inputs: Users and items cluster assignments, training data points, N_u, and randomly initialized parameters.
% Outputs: Trained parameters.
for $count = 1, ..., \#Iterations$ **do**
  for $u = 1, ..., n$ **do**
    %Computes the components independent of i
    $p'_u \leftarrow T_\alpha(p_u, p_{C_u}) + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} T_\alpha(y_j, y_{C_j}) + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_u - b_j) T_\alpha(x_j, x_{C_j})$
    $sum \leftarrow 0$
    for $i \in N(u)$ **do**
      $r_{ui} = p'_u . T_\alpha(q_i, q_{C_i})^T + T_\beta(b_u, b_{C_u}) + T_\beta(b_i, b_{C_i})$
      $e_{ui} = r_{ui} - r_{ui}$
      %Accumulate information for gradient steps
      $sum \leftarrow sum + e_{ui} . T_\alpha(q_i, q_{C_i})$
      % Perform gradient steps
      $q_i \leftarrow q_i + \gamma(e_{ui} . T_\alpha(p_u, p_{C_u}) - \lambda . q_i)$
      $q_{C_i} \leftarrow q_{C_i} + \gamma(e_{ui} . T_\alpha(p_u, p_{C_u}) - \lambda . q_{C_i})$
      $b_u \leftarrow b_u + \gamma(e_{ui} - \lambda . b_u)$
      $b_i \leftarrow b_i + \gamma(e_{ui} - \lambda . b_i)$
      $b_{C_u} \leftarrow b_{C_u} + \gamma(e_{ui} - \lambda . b_{C_u})$
      $b_{C_i} \leftarrow b_{C_i} + \gamma(e_{ui} - \lambda . b_{C_i})$
    end for
    for $i \in N(u)$ **do**
      %Perform gradient steps
      $x_i \leftarrow x_i + \gamma |N(u)|^{-\frac{1}{2}} . (r_{ui} + T_\beta(b_u, b_{C_u}) + T_\beta(b_i, b_{C_i})) . sum - \lambda x_i$
      $x_{C_i} \leftarrow x_{C_i} + \gamma |N(u)|^{-\frac{1}{2}} . (r_{ui} - T_\beta(b_u, b_{C_u}) + T_\beta(b_i, b_{C_i})) . sum - \lambda x_{C_i}$
      $y_i \leftarrow y_i + \gamma |N(u)|^{-\frac{1}{2}} . sum - \lambda y_i$
      $y_{C_i} \leftarrow y_{C_i} + \gamma |N(u)|^{-\frac{1}{2}} . sum - \lambda y_{C_i}$
    end for
  end for
end for

3.3 Experiment Results

We set up our experiment on two well-known recommendation datasets to validate our proposed recommendation models. The MovieLens100k data set was collected by the GroupLens Research Project at the University of Minnesota. It contains 100,000 ratings from 943 users on 1,682 movies where each user has rated at least 20 movies [17]. The package includes five randomly 80%/20% splits of dataset into training and test sets. We employ these training and test sets provided in the package (u1, u2, .., u5) in our evaluation. The Netflix dataset contains over 100 million ratings from 480,189 users who have rated 17,770 movies. In both datasets, ratings are in a range of $[1, 5]$. Both datasets are very sparse as we know only 1% of ratings and 99% of ratings are unknown. We run each algorithm five times on these datasets to remove the impact of random initializations on the experimental results. Thus, our reported results are the averaged result of these five runs.

In the following subsections, we first describe our clustering process in Section 3.3.1. We then employ these found clusters in our proposed methods and evaluate their rating prediction accuracy (Section 3.3.2). In Section 3.3.3, we evaluate the impact of our proposed clustering-based method in rating prediction accuracy for cold-start users. Finally, in Section 3.3.4, we present additional experiments that evaluates the impact of the quality and the quantity of clusters on improving rating prediction accuracy.

3.3.1 Clustering Users And Items

The rating matrix R is very large and sparse. Thus, clustering this matrix would be costly. Therefore, we first apply matrix factorization on matrix R to reduce the dimension of user space and item space. Then, we cluster users and items separately in these generated spaces in a much lower cost. We start by applying biased matrix factorization on both datasets to find their users' and items' latent vectors. These latent vectors (learned on the train sets) are then used for the clustering purpose. It is expected that items (users) with similar latent

Table 3.1: The final selected m' and n' that is employed in the evaluation of our extension methods. These numbers are found employing a validation set from each dataset, and by trying different selection of m' and n'.

Dataset	n'	m'
MovieLens100k	100	100
Netflix	1000	500

vectors are similar in reality as well. In [26], it is shown that similar latent vectors represent similar movies in the Netflix datasets. We try four different clustering methods, and as shown in Section 3.3.4, K-Means results in the best prediction accuracy in both of these datasets. Expectation maximization achieves almost the same rating prediction accuracy but with much more complexity.

We vary different selections of possible clusters by changing the number of clusters (m', n'). Finally, by trying different sizes of clusters on the proposed models, the number of clusters that achieves smallest RMSE (higher accuracy) on the validation set will be selected as the best choice of m', and n'. The final selected m's and n's that are employed in the evaluation of our extension methods are presented in Table 3.1. Based on our experiment, the clusters should not be too broad or too small. Figure 3.3 illustrates the distribution of clusters of items and users with different numbers of members in the Netflix dataset. It seems our final selected number of clusters (Figures 3.3.e and 3.3.f) achieves a normal-like distribution with fewer too large or too small clusters.

Table 3.2 shows the movies inside a number of found clusters in the Netflix dataset. As shown, it seems that movies in the same genre and almost similar years of production tend to be in the same clusters. For instance, 'category 295' includes different versions of 'Lord of the Rings' and 'Star Wars' movies accompanied by a number of other movies in 'Adventure' and 'Fantasy' genres[1] such as 'The Matrix'. 'category 344' also contains different versions of 'X-Men', 'Spider Man' and 'Harry Potter' movies. Or, 'category 420' includes a number of documentaries and live concerts. As no information about the users is provided, the clusters

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

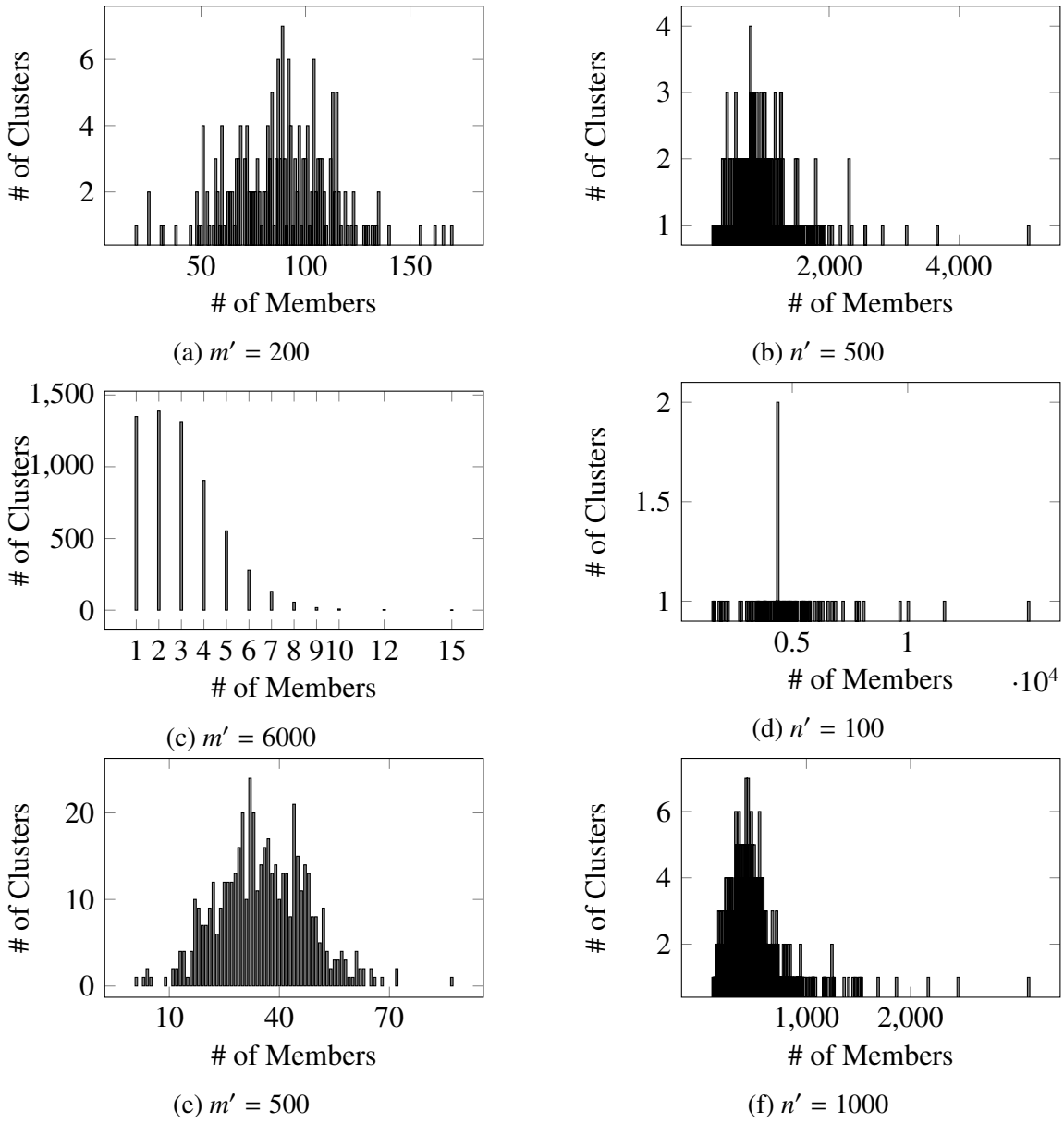


Figure 3.3: Distribution of clusters of items and users in different sizes in the Netflix dataset.

of users cannot be judged. Let's remember that the movies' names (identities) are not used anywhere in this experiment; these names are only employed for better demonstration of the clusters. Obviously the clusters are not perfect and there are always items that are wrongly clustered. Evaluating the clusters is subjective and depends on the view of the evaluator. However, based on the our point of view at least two thirds of the clustered items seem meaningful.

3.3.2 Comparison Regarding Rating Prediction

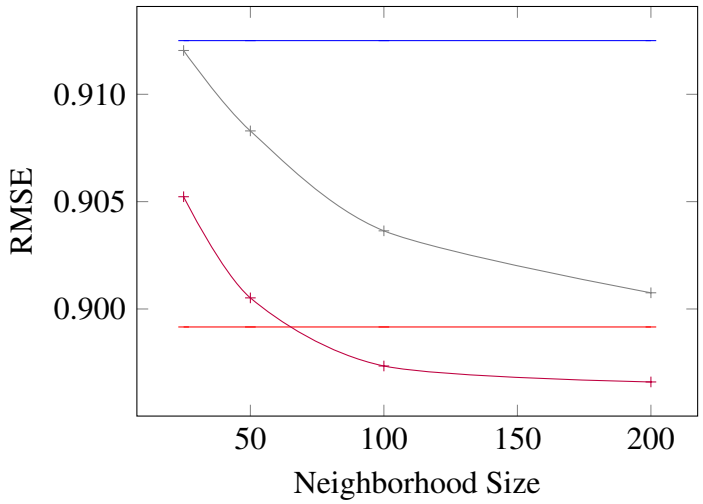
In our proposed clustering-based methods, two variables α and β are used to control the balance between the traditional user-item and the cluster-level rating prediction levels. For $\alpha = \beta = 0.0$, this model does not consider the impact of cluster-level predictions. Thus, the resulting RMSE is similar to the accuracy of the traditional model as expected. A validation set is used to determine the best selection of these variables in our experiment. Table 3.3 illustrates the RMSE result of CBMF model by selecting different combinations of α and β in the MovieLens dataset. $\alpha = 0.5$ and $\beta = 0.8$ is selected as the final values in our experiment in this dataset.

Figure 3.5 illustrates a comparison between the RMSE results of the biased matrix factorization and our proposed clustering-based matrix factorization for different selections of l. As shown, CBMF outperforms the biased matrix factorization even when BMF employs higher l. Additionally, Figure 3.4 shows a comparison between the accuracy of four models: Biased Matrix Factorization (BMF), Asymmetric SVD (Asysvd) [23], and our clustering-based extensions of them. It shows that CBMF achieves a better RMSE compared with the non-extended neighbourhood-aware model (Asysvd) when this model does not employ enough large numbers of neighbors. Neighbourhood-aware models are usually sensitive to the selection of the neighbourhood size. By employing larger neighbourhood size, the RMSE result is improved. However, this practice results in higher complexity and lower comprehensibility of these models.

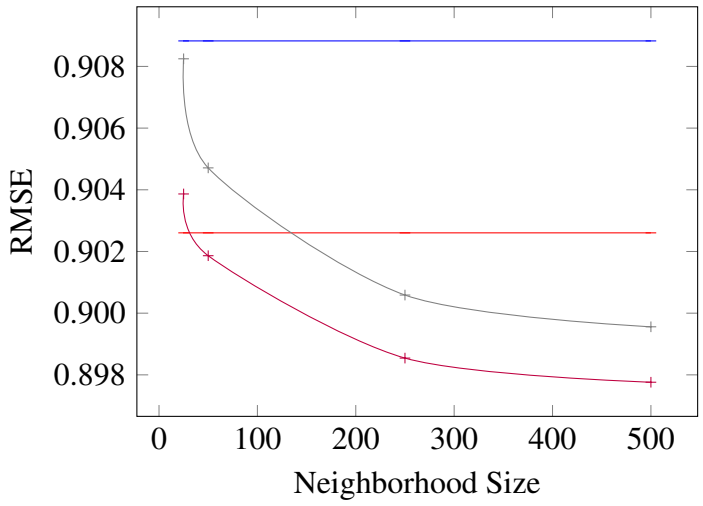
We apply a t-test to ensure the significance of our results. We train the extended and non-extended models several times with different initialized parameters. By employing t-tests on

Category ID	Movie Title (Production Year)
cluster295	Lord of the Rings: The Return of the King: Extended Edition: Bonus Material (2003), Lord of the Rings: The Two Towers: Extended Edition (2002), Firefly (2002), Raiders of the Lost Ark (1981), Star Wars: Clone Wars: Vol. 1 (2004), Star Wars: Episode VI: Return of the Jedi (1983), Lord of the Rings: The Two Towers: Bonus Material (2002), Lost: Season 1 (2004), Aliens: Collector's Edition: Bonus Material (1986), Batman Begins (2005), Lord of the Rings: The Two Towers (2002), Lord of the Rings: The Fellowship of the Ring: Bonus Material (2001), Lord of the Rings: The Return of the King: Bonus Material (2003), Battlestar Galactica: The Miniseries (2003), Crouching Tiger (2000), Star Wars: Episode IV: A New Hope (1977), Lord of the Rings: The Fellowship of the Ring (2001), Band of Brothers (2001), Battlestar Galactica: Season 1 (2004), Star Wars: Episode V: The Empire Strikes Back (1980), The Matrix (1999), The Indiana Jones Trilogy: Bonus Material (2003), Lord of the Rings: The Return of the King: Extended Edition (2003), The Lord of the Rings: The Fellowship of the Ring: Extended Edition (2001), Indiana Jones and the Last Crusade (1989), Lord of the Rings: The Return of the King (2003), Star Wars Trilogy: Bonus Material (2004)
cluster344	Pirates of the Caribbean: The Curse of the Black Pearl: Bonus Material (2003), X2: X-Men United: Bonus Material (2003), X-Men (2000), Harry Potter and the Prisoner of Azkaban (2004), Harry Potter and the Sorcerer's Stone (2001), Monsters (2001), Harry Potter and the Chamber of Secrets (2002), X2: X-Men United (2003), The Incredibles: Bonus Material (2004), Harry Potter and the Prisoner of Azkaban: Bonus Material (2004), Spider-Man 2 (2004), X-Men: Bonus Material (2000), Beauty and the Beast: Special Edition: Bonus Material (1991), Harry Potter and the Chamber of Secrets: Bonus Material (2002), Harry Potter and the Sorcerer's Stone: Bonus Material (2001), Farscape: The Peacekeeper Wars: Bonus Material (2004), Spider-Man (2002), Pirates of the Caribbean: The Curse of the Black Pearl (2003)
cluster420	ABC Primetime: Mel Gibson's The Passion of the Christ (2004), Jay Jay the Jet Plane: Good Friends Forever (2003), Lassie: The 50th TV Anniversary Collector's Edition (1954), Jesus and His Times (2000), Jesus of Nazareth (1977), Wiseguy: Between the Mob and a Hard Place (1989), Jesus and the Shroud of Turin (1999), The Bible Collection: Moses (1996), National Geographic: Inside American Power: The White House (1996), Barney: Come on Over to Barney's House (2000), Chicago: AE Live by Request (2003), Love Comes Softly (2003), The Commish: Season 1 (1991), Oliver Twist (1999), The Ultimate National Geographic WWII Collection (2004), Twin Towers (2003), The Gospel of John (2003), Hans Brinker (1962), Lorna Doone (2000), The Miracle Maker: The Story of Jesus (2000), Jeremiah: The Bible (1998), Thomas and Friends: Calling All Engines (2005), The Bible Collection: Jacob (1994), New York Firefighters: The Brotherhood of 9/11 (2002), Parineeta (2005), Genesis: The Way We Walk: Live in Concert (2001), Billy Joel: The Essential Video Collection (2001), National Geographic: Ambassador: Inside the Embassy (2002), Joshua (2002), The Barkleys of Broadway (1949), Michael W. Smith: Live in Concert: A 20 Year Celebration (2004), National Geographic: Inside American Power: Air Force One (2001), Bear in the Big Blue House Live! (2002), Piano Grand: A Smithsonian Celebration (2000), Samantha: An American Girl Holiday (2004), Chris Botti and Friends: Night Sessions: Live in Concert (2002), Denise Austin: Personal Training System (2004), Walt: The Man Behind the Myth (2001), The Story of Jesus for Children (1979), Joseph: King of Dreams (2000), In Old Chicago (1937), Visions of Greece (2002)

Table 3.2: The table shows the movies inside a number of formed clusters in the Netflix dataset. As shown, it seems that movies in same genre and almost similar years of production tend to be in same clusters. Careful analysis shows that about 2/3 of the clusters have some meaningful similarities.



(a) MovieLens



(b) Netflix

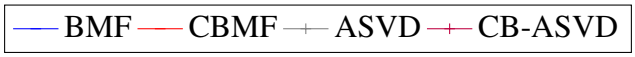


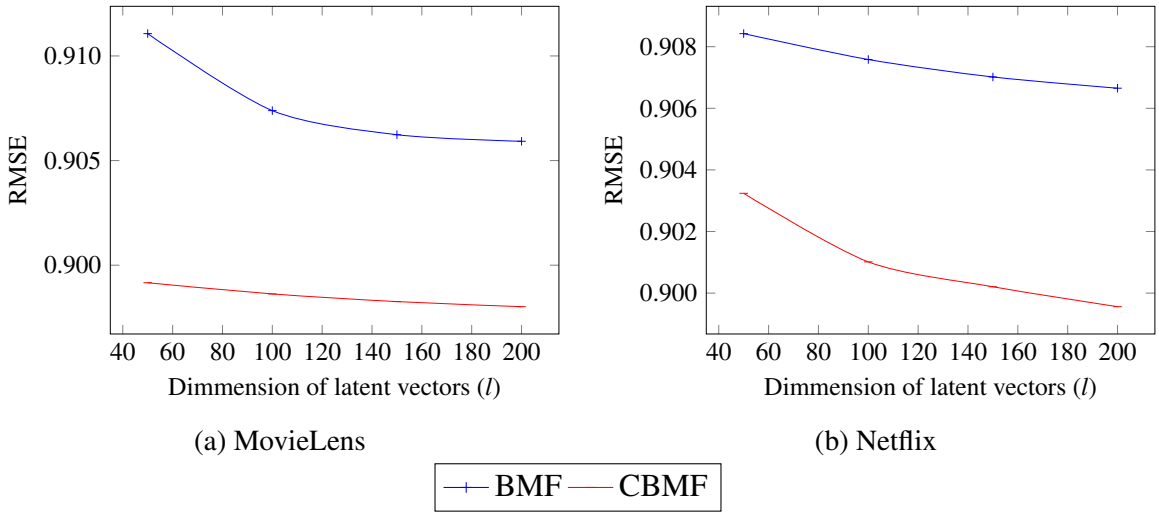
Figure 3.4: The accuracy of the proposed clustering-based models applying on the two datasets. It shows that our proposed extensions outperform their non-extended models in the both datasets ($l = 50$ is used to achieve these results).

α / β	0	0.2	0.4	0.6	0.8	1
0	0.922	0.921	0.921	0.920	0.919	0.931
0.2	0.914	0.914	0.913	0.912	0.911	0.921
0.4	0.913	0.913	0.912	0.911	**0.909**	0.920
0.6	0.914	0.914	0.913	0.912	0.911	0.922
0.8	0.918	0.917	0.917	0.916	0.918	0.933
1	0.927	0.927	0.927	0.928	0.931	1.072

Table 3.3: RMSE results from applying CBMF with different values for α and β on a validation set in the MovieLens dataset.

n' / m'	10	50	100	150	200
10	0.9140	0.9115	0.9106	0.9113	0.9114
50	0.9140	0.9107	0.9105	0.9109	0.9110
100	0.9135	0.9101	**0.9097**	0.9101	0.9104
150	0.9112	0.9110	0.9109	0.9112	0.9145
200	0.9144	0.9113	0.9115	0.9112	0.9116

Table 3.4: Resulting RMSE by applying CBMF with different values of clusters (m' and n') on a validation set in the MovieLens dataset.



(a) MovieLens (b) Netflix

Figure 3.5: A comparison between Biased Matrix Factorization (BMF) and our proposed Clustering-based Matrix Factorization (CBMF) for different selection of l (dimension of latent vectors).

the achieved RMSE results, for $df = 30$ and with probability of 90%, the mean of our extended methods' rating predictions is lower than the mean of their non-extended models' rating predictions. Table 3.5 shows these selected values that we employ in the training process of our proposed CBMF model in our experiment in the datasets (Algorithm 3).

3.3.3 Cold-Start Users

Cold-starts are users who newly enters into a recommendation system and they do not have any or more than few ratings in the system. Consequently, the system is unable to locate their preferences adequately. Cold-start users are one of the major challenges in recommendation systems [26], [22], [42]. Using the neighbourhood information is a common solution to im-

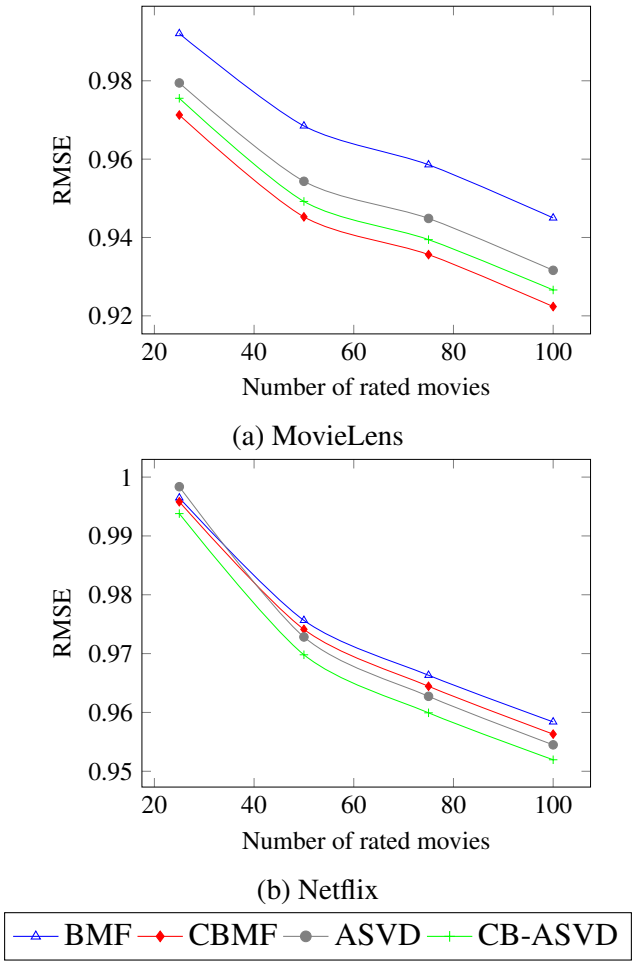
Parameter	Netflix	MovieLens
n'	1000	100
m'	500	100
α	0.1	0.5
β	0.7	0.8
γ_1	0.005	0.005
γ_2	0.002	0.005
γ_3	0.005	0.005
γ_4	0.0005	0.00002
λ_1	0.01	0.035
λ_2	0.1	0.065
λ_3	0.0001	0.0001
λ_4	0.055	0.0001

Table 3.5: Employed parameters in Algorithm 3 in the datasets.

prove recommendation accuracy for these users. Our produced coarse matrix R_C reduces the sparsity of rating matrix R. Thus, R_c can generalize any minor feedback from cold-start users to improve their related recommendation accuracy.

For example, assume that a cold-start user u has rated movie "Lord of The Rings". Using neighborhood information we can only find out that user u may like similar movies such as "Star Wars". However, many similarities between movies and users will be ignored because of the sparsity of rating matrix R. On the other hand, our clustering level modeling of movies considers a higher level of similarities between movies. Thus, our R_C can extract more information from any minor feedback from cold-start user u. Once these clusters are formed, known ratings in the clusters can help to predict the unknown ratings for cold-start users.

Cold-start users are examined separately in our experiment to see if our extension models improve the accuracy of their recommendations. Figure 3.6 illustrates the RMSE results of our proposed model for cold start users in both datasets. The horizontal axis represents the increase of the known ratings for users. As shown, our extended models effectively improve the prediction accuracy for cold-start users. For instance, in the MovieLens dataset, CBMF model results in an almost 0.97 of RMSE for user with fewer than 25 known ratings, which is as good as the RMSE result of BMF model for users who have fewer than 50 known ratings.



(a) MovieLens

(b) Netflix

Figure 3.6: A comparison over the RMSE of the extended and non-extended models for the cold-start users.

3.3.4 Sub-experiments

In this section we explain two sub experiments; We first try four different clustering algorithms to examine the impact of the clusters' quality on our proposed model prediction accuracy (Section 3.3.4). Using this experiment we also select the clustering algorithm that achieve the best prediction accuracy as our default clustering method. We then vary the number of clusters in the selected clustering algorithm to create clusters in different sizes. In Section 3.3.4, we show that adding more clusters in variety of sizes will improve our proposed model's prediction accuracy further. Thus, the following experiments evaluate the impact of the quality and the quantity of clusters on improving rating prediction accuracy in our proposed model.

Different Clustering Methods

In this section, we employ four different clustering methods to examine the impact of clustering quality on our proposed model's rating prediction results. We apply following clustering methods on the generated Ps and Qs in both datasets:

- **K-Means:** It is based on partitioning data into K clusters that each data point belongs to the cluster with the nearest mean. This is a well-known clustering method that achieves a high quality of clusters.

- **Expectation Maximization (EM):** It is an expensive but high quality clustering algorithm. EM models the data points with a fixed number of Gaussian distributions, those are initialized randomly and their parameters are iteratively optimized to fit better to the data points.

- **Density-Based K-Means:** It is a fast version of K-Means that tries to find the clusters based on the density of data points in a region.

- **Farthest First:** It is another fast version of K-Means that in each step places each cluster center at the point farthest from the existing cluster center. This process results in less

readjustments and greatly speed up the clustering model. However, it mostly achieves a lower quality of clustering.

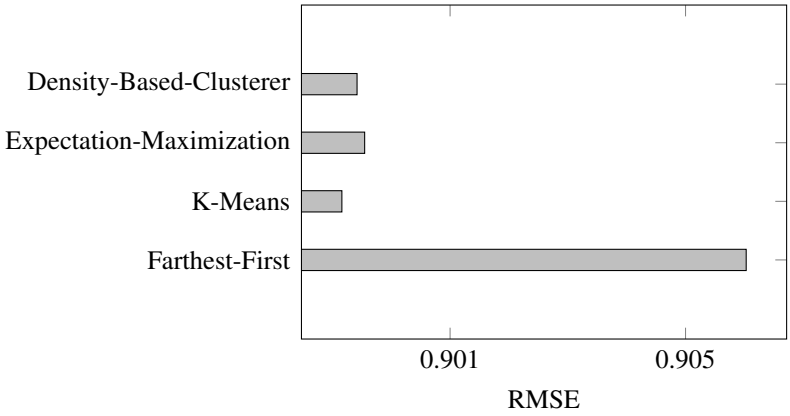
By employing the found clusters from these four clustering methods in our CBMF model, we examine the impact of clustering quality on CBMF's resulting RMSE (Figure 3.7). As expected, Farthest First model achieves lower accuracy as it provides a poor clustering performance. Density-Based K-Means also is a fast version of K-Means which provides a slightly faster but lower quality of clustering. Thus, Density-Based K-Means shows a lower rating prediction accuracy compared to the original K-Means. Also, in both datasets, EM and K-Means result in almost the same rating prediction accuracy. However, EM has more complexity than K-Means. Thus, we select K-Means as the default clustering method in our experiment.

In addition, CB-Asysvd model results in a RMSE of 0.90523 for a random assignment of the clusters ($m' = n' = 100$) in the MovieLens100k dataset, while it achieves 0.89668 by employing K-Means's found clusters. Therefore, by increasing the quality of clusters, our proposed CBMF model achieves a better rating prediction accuracy.

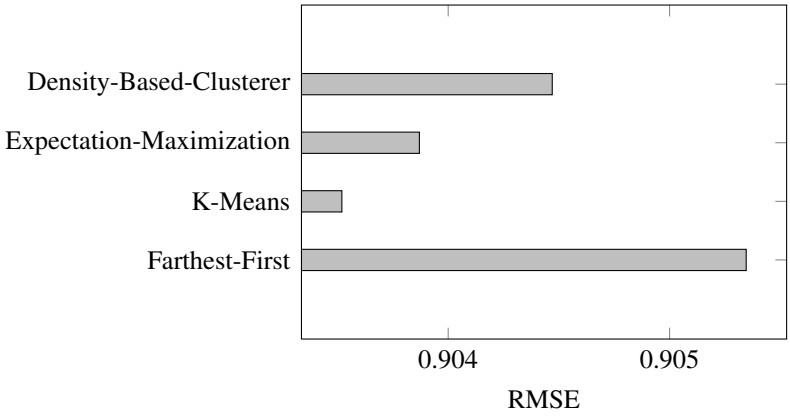
Employing More Clusters

As mentioned in Section 3.2.2, different numbers of clusters capture different levels of similarity between users and item. For instance, by clustering movies into 10 clusters in the MovieLens dataset, we expect to find similar movies based on a general feature such as genre. However, by increasing the number of clusters to 200, we expect to distinguish between movies based on more concrete differences such as the production year. Thus, we expect to improve rating prediction accuracy further by adding these clusters with variety of sizes into the extended version of our proposed CBMF model (Equation 3.8), because they capture different levels of informative similarities among items and users. For making these clusters in the MovieLens dataset, we apply K-Means on users and items latent spaces to find 10 clusters $n' = m' = 10$, we then gradually increase m' and n' until $n' < n/4$ and $m' < m/4$.

As Figure 3.8 shows by employing more clusters in variety of sizes, rating prediction ac-



(a) MovieLens



(b) Netflix

Figure 3.7: Applying different clustering methods in users and items latent spaces in both datasets and its effect on the CBMF's result.

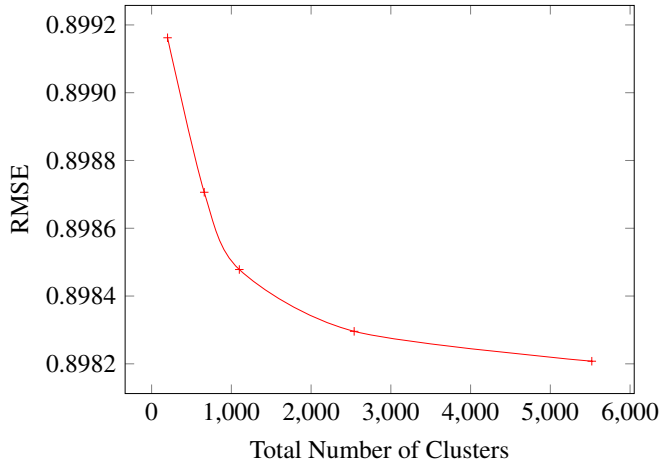


Figure 3.8: Applying clustering multiple times with different number of clusters and employ those found clusters in CBMF model. By employing more clusters with variety of sizes, rating prediction (RMSE) improves slightly.

curacy improves in the MovieLens dataset. However, we do not use all these clusters in our final CBMF model because they add more complexity on the methods and improve the rating prediction accuracy slightly.

3.4 Complexity

Both CB-MF and CB-ASVD models have the same time and space complexity as their non-extended models. This is because we employ almost the same training model as those non-extended models but once for the traditional ratings and another time for the cluster-level ratings. Thus, our extended models have almost twice of training time compare to MF and ASVD but same complexity. Although to produce the course matrix, a preprocessing complexity will be added to our proposed models to first factorizing the traditional rating matrix and then clustering the generated latent vectors.

3.5 Relation to Previous Work

Employing clustering to categorize collaborative information, and using these clusters to predict the unknown preferences has been employed before in a number of works such as [54], and [15]. However, the common preferences between the clusters have been less considered. For instance, Gueye et al. [15] tries to add the effect of clusters in biased matrix factorization model. However, they limit themselves to use the advantage of clusters' biases only. The predictor function that they use is as follows:

$$\hat{r}_{ui} = p_u.q_i^T + \mu_{C_{G(i)}} + b_{u,C_{G(i)}} + b_i$$

where $C_{G(i)}$ is a function, which returns for any item i its group, $\mu_{C_{G(i)}}$ is the average ratings in $C_{G(i)}$, and $b_{u,C_{G(i)}}$ is the bias of user u for the group of items C_{G_i}. George et al. in [14] use the same technique to improve the rating prediction accuracy by employing the rating averages for users, items, user-clusters, and item-clusters. However, as the common interests between clusters are not considered in these papers, it is expected that our method results in a better rating prediction accuracy. For instance, running 'CBMF' considering the clusters' biases only achieves the RMSE of 0.907765 in the MovieLens100k dataset, which is not better than the RMSE of our 'CBMF' method using the same parameters.

Jamali et al. [20] and Beutel et al. [3] propose promising probabilistic models to co-cluster users and items and then consider their cluster-level preferences to improve the rating prediction accuracy. However, because of the expensive nature of probabilistic modeling they have to employ heuristics to reduce both the learning time and the consumed memory. However, our proposed model is much simpler in its nature and like a general wrapper can be applied to all collaborative filtering methods.

In addition, Xu et al. [54] employ the clusters of users and items in different ways to improve the accuracy of predictions. They cluster items and users into subgroups where each user or item can belong to more than one cluster. Their main idea is to apply number of col-

laborative filtering algorithm in each subgroup and then merge the prediction results together. However, they have to consider clusters reasonably larger to possess enough known ratings in each subgroup for the learning process. Considering only high-level clusters leads to miss many dissimilarities between users and items (As discussed in Section 3.2.2). [10] presents a complete survey on neighbourhood-based recommendation methods that covers many other extensions on matrix factorization.

Chapter 4

Improving Top-N Recommendation for Cold-Start Users via Cross-Domain Information

4.1 Introduction

As discussed in Section 1.3, collaborative filtering based recommendation systems employ observed ratings to generate a list of relevant items for each user. Netflix, Amazon, and YouTube are examples of large companies that have successfully integrated collaborative filtering in their recommendation engines. Although these models do not achieve good recommendations for cold-start users [26, 42, 46].

In the last few years, cross-domain recommendation systems (Section 1.6) have been employed to include available information from users in other domains to improve recommendation accuracy [8, 28, 29, 51]. However, most of these proposed models focus on improving rating prediction accuracy, often observed in terms of the root mean square error (RMSE). Although the major role of a recommendation system is to make a short list of relevant items for each user. Hence, rating prediction for all non-observed ratings is definitely a less important

task than accurately ranking the top relevant items as a short list of recommendations, often measured by top-N recommendation task. Thus, optimizing the recommendation model regarding RMSE is highly criticized [9, 23, 49, 50] as it does not necessarily improve the top-N recommendation task.

In addition, it is empirically proved that because of unbalanced distribution of observed ratings, employing unobserved ratings as negative feedback can improve top-N recommendation tasks dramatically (Section 2.0.6). For instance, Steck in [49] shows that employing unobserved ratings in matrix factorization achieves 64% of recall in the well-known Netflix dataset. However, both matrix factorization excluding unobserved ratings and the well-known integrated model proposed in [23] result in 39% and 43% recall respectively. In a this chapter, we take advantage of unobserved ratings on two levels: the traditional user-item level and our proposed cluster level of users and items in a latent space [34].

In Chapter 3, we define cluster-level 'coarse' matrices for *single domains*. These coarse matrices capture the shared interests among the cluster of users and the cluster of items. Thus, it generalizes the preferences of users on items (into a cluster level rating matrix) to reduce the sparsity of the original rating matrix. In this chapter, we extend our previous work from single-domain into cross-domain recommendation systems by employing the partially overlapped users and items between multiple domains to transfer their cluster-level preferences as the auxiliary sources of information.

In sum, this new method has several novel contributions:

- We utilize the information of unobserved ratings in cross-domain recommendation systems via a cluster-level space.

- Cross-domain methods mostly need heavy computations to find a tranferring function between each pair of domains [8, 29, 51], while our proposed method simply transfer the knowledge of several auxiliary domain between each other in one step (Section 4.2.1).

- We practically show that integrating the transferred cluster-level and the traditional trans-

ferred user-item level knowledge can significantly improve the recommendation quality.

To validate our new method, we set up our experiment on two datasets: the Amazon dataset and the Epinions dataset. Both datasets include multiple domains such as DVD, Video, Electronics, etc. Our experiments show that our proposed cross-domain clustering-based matrix factorization model significantly increases the recall in top-N recommendation for all users, and cold-start users in particular. For example, our method achieves a recall of 43% on average for all users compared to 39% using the previous methods in the Amazon dataset. We also observe 25% improvements of top-N recommendation in the Epinions dataset. For cold-start users, our method improves recall to 21% on average, whereas previous methods result in only 15% recall by including data from other domains (see Section 4.3 for more details) in the Amazon dataset. We observed almost same improvements in Epinions dataset as well. Note that it is often difficult to make even a small improvement in recommendations, and for cold-start users in particular. For instance, the difference between the biased matrix factorization (Section 2.0.2) and the well-known Integrated model [23] (Section 2.0.6) was only 3% of recall for top-N recommendation tasks in the Netflix dataset. Hence, our improved rate of recall is quite significant.

4.2 The Proposed Method

As mentioned in Chapter 1, in a general collaborative filtering based recommendation system, we have a set of users $U = \{u_1, u_2, \ldots, u_n\}$ and a set of items $I = \{i_1, i_2, \ldots, i_m\}$ that are accompanied by a rating matrix $R = [r_{ui}]_{n \times m}$ where r_{ui} represents the rating of user u on item i. A collaborative filtering based recommendation system consists of making a short list of relevant items to user u based on the ratings inside rating matrix R. However, the quality of this rating prediction task is sensitive to the number of observed ratings form user u, N_u. Hence, collaborative filtering based recommendation systems are unable to generate accurate recommendations for users who have made no or very few observed ratings, known as cold-start

users. Consequently, more information is needed to achieve a better quality of recommendations for cold-start users.

A major source of extra information is the ratings that these users have made in the other domains. For example assume an e-store website with different departments including 'books', 'movies', 'computers', etc. User u may have many observed ratings in the 'books' domain, but no ratings in the 'movies' domain. Thus, a natural solution is using observed ratings from the 'books' domain to generate a better list of recommendations in the 'movies' domain for user u. As discussed in Section 1.6, this solution is called cross-domain recommendations for a set of domains $D = \{d_1, d_2, \ldots, d_t\}$. These auxiliary domains can be categorized according to their users and items overlap, from full-overlap, users-overlap and items-overlap domains, to no-overlap domains [8]. Our proposed cross-domain recommendation model is based on a partial overlap of users and/or items between the target domains and the auxiliary domains.

In Chpater 2, we define cluster-level 'coarse' matrices in *single domains*. These coarse matrices generalize the relation of users and items into a cluster-level by capturing the mean rating between the cluster of users and the cluster of items. In Section 4.2.1, we extend our previous work to cross-domain recommendation systems. In a cross-domain scenario, we find the relations among these clusters in different domains. Thus, we can predict the preferences of the users in a target domain by employing their cluster level preferences in the auxiliary domains.

4.2.1 Making A Cross-Domain Coarse Matrix

As discussed earlier, cross-domain auxiliary information can be employed in the recommendation model to increase the recommendation accuracy further. In this chapter, we build a cross-domain cluster-level 'coarse' matrix R^c, which captures the shared interests among the cluster of users and the cluster of items between multiple domains. Figure 4.1 illustrates rating matrix R including ratings from Music and Movies domains. As shown, it is assumed that these two domains have partially overlapped users (dashed area). Let's assume that user u is a shared

user between these domains. User u may rate item i in Music domain and item i' in Movies domain. In a classic matrix factorization, only these shared ratings will be transferred between the two domains. However, all the entries in the top right and lower left (the white areas in Fig 4.1:Left) are missing values, and thus the rating matrix is too sparse.

In our model we propagate these shared ratings into a cluster level (Figure 4.1:Right). Hence, we reduce the sparsity of rating matrix R by propagating the observed ratings into unobserved ratings in coarse matrix R^c. In Figure 4.1 for instance, we may propagate the individual interest of users inside cluster 4 in couple of cartoons into a cluster level interest from cluster 4 in the cluster of cartoons, as new entities. Note that the white area (missing values) is much reduced in Figure 4.1 (right side). As overlapped users are separately clustered in each domain, they belong to more than one cluster in a cross-domain scenario. By factorizing this new matrix, coarse matrix, we will have cluster-level preference prediction for each cluster of users.

Here is a formal description of our proposed method. Let's assume $C_{U,u}^{d_j}$ as the uth cluster of users in domain d_j, and $C_{I,i}^{d_j}$ is the ith cluster of items in domain d_j. After finding the domain-specific clusters, we define cross-domain coarse matrix R^c as follows:

$$R^c = \begin{bmatrix} R^c_{d_1,d_1} & R^c_{d_1,d_2} & \cdots \\ R^c_{d_2,d_1} & R^c_{d_2,d_2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}, \tag{4.1}$$

where:

$$R^c_{d_j,d_{j'}} = \begin{bmatrix} r_{C_{U,1}^{d_j} C_{I,1}^{d_{j'}}} & r_{C_{U,1}^{d_j} C_{I,2}^{d_{j'}}} & \cdots \\ r_{C_{U,2}^{d_j} C_{I,1}^{d_{j'}}} & r_{C_{U,2}^{d_j} C_{I,2}^{d_{j'}}} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}, \tag{4.2}$$

and each $r_{C_{U,u}^{d_j} C_{I,i}^{d_{j'}}}$ is defined as follows:

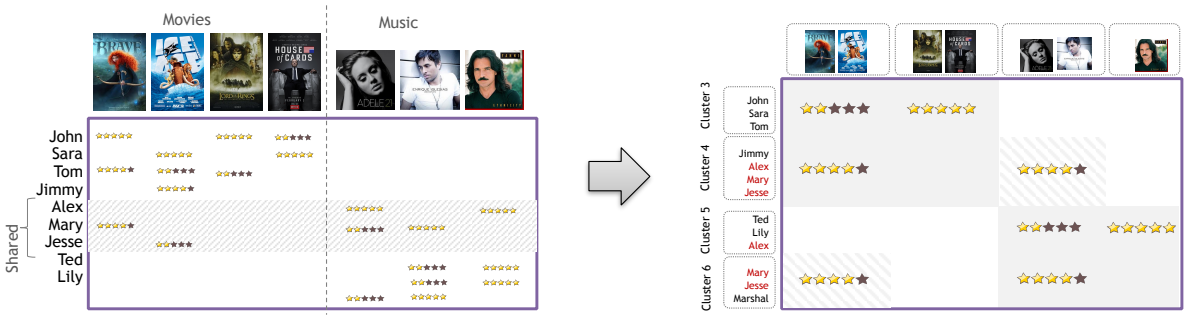


Figure 4.1: (Left) Cross-Domain rating matrix R including rating matrices of domains Music and Movies with overlapped users (dashed area). The rating matrix is very sparse as many entries in the top right and lower left are missing values. (Right) Coarse matrix R^c including mean ratings between cluster of users and cluster of items. As shown, the coarse matrix reduces the sparsity of R by propagating the observed ratings into unobserved ratings. Note that the white area (missing values) is much reduced.

$$r_{C_{U,u}^{d_j} C_{I,i}^{d_{j'}}} = \begin{cases} \dfrac{\sum_{u',i'} r_{u'i'}}{N(C_{U,u}^{d_j}, C_{I,i}^{d_{j'}})} & N(C_{U,u}^{d_j}, C_{I,i}^{d_{j'}}) > thr \\ \text{unobserved} & N(C_{U,u}^{d_j}, C_{I,i}^{d_{j'}}) \leq thr \end{cases} . \tag{4.3}$$

$i' \in C_{I,i}^{d_{j'}}$ and $u' \in C_{U,u}^{d_j}$; $N(C_{U,u}^{d_j}, C_{I,i}^{d_{j'}})$ is the number of observed ratings that the users inside cluster $C_{U,u}^{d_j}$ have made on the items inside cluster $C_{I,i}^{d_{j'}}$. Let's remember that each overlapped user or item belongs to more than one cluster because they are clustered separately inside two or more different domains. For example, assume that user u belongs to $C_{U,u}^{d1}$ in domain $d1$, and also belongs to cluster $C_{U,u}^{d2}$ in domain $d2$. In this chapter, we suppose that $r_{C_{U,u}^{d1} C_{I,i}^{d}} = r_{C_{U,u}^{d2} C_{I,i}^{d}}$ (Figure 4.1). As is shown in Figure 4.1, this assumption propagate observed ratings into unobserved ratings. Thus, cluster-level rating matrix R^c (Figure 4.1:right) reduces the sparsity of R (Figure 4.1:left). However, sometimes there is not enough evidence to support this propagation. Hence, we employ a fixed threshold value thr to remove low confident relations between the clusters (we take $thr = 5$ in our experiment).

4.2.2 Generating Recommendations

Later in Section 4.2.3, we will show how to factorize matrices R, and R^c to rank the unobserved ratings. Here, we will describe the way that we make the cluster-level recommendations and how we aggregate them with traditional user-item level recommendations. Let's define ${N_u}^C$ as the number of clusters (in different domains) that user u belongs to, ${N_i}^C$ as the number of clusters (in different domains) that item i belongs to, predicted rating matrix $\hat{R} = [\hat{r}_{ui}]$, and its cluster-level predicted rating matrix $\hat{R}^c = [\hat{r}^c_{ui}]$, where $\hat{r}_{ui} = r_m + P_u Q_i^T$, and:

$$\hat{r}^c_{ui} = r_m + \frac{\sum_{d \in D} P^c_{C^d_{U,u}} Q^c_{C^d_{I,i}}}{N^C_u . N^C_i}. \tag{4.4}$$

The cluster-level predicted ratings in $\hat{R}^c$ are too general to be used solely. Hence, we integrate these two matrices linearly to achieve our final predictions, as follows:

$$R^* = \alpha \hat{R}^c + (1 - \alpha)\hat{R}, \tag{4.5}$$

where $\alpha \in [0, 1]$ is a fixed tuning parameter, and optimized via cross-validation. Thus, we employ matrix $R^* = [r^*_{ui}]$ to rank relevant items to each user. For evaluating these recommendations, we use the top-N recommendation metric which is proposed by Koren in [23].

4.2.3 Factorizing Matrices Considering Unobserved Ratings

Usual collaborative filtering based recommendation models are based on observed ratings. However, Steck shows [49] that the distribution of usual datasets in recommendation systems are unbalanced and any unobserved ratings can be considered as low confidence non-preference feedback from users. Thus, those observed-ratings based algorithms ignore much useful feedback from users. Steck also shows that by considering these unobserved ratings as a low rating value, r_m (such as $r_m = 2$ or any other value lower than the mean of observed ratings), the accuracy of recommendations increases dramatically. For example, he empirically

shows that highly complex methods such as proposed integrated method [23] achieves a recall of 42% based on top-N recommendation (N=20) in the well-known Netflix dataset, but his unobserved-ratings integrated model increases this number to 64%.

However, factorizing a full-filled rating matrix (filled by replacing all unobserved ratings with r_m) with many number of users and items will be computationally expensive. Hence, he proposed a new Alternative Least Squares (ALS) based learning model to factorize this full-filled rating matrix into matrices P and Q more efficiently, which is described in more details in Section 2.0.6.

We extend Steck's approach by applying this factorizing technique in two levels:

- To factorize cross-domain users-items level rating matrix R.
- To factorize cross-domain cluster level rating matrix R^c (Equation 4.1).

Thus, to incorporate unobserved ratings to our method, we employ similar learning steps as Equation 2.14 and Equation 2.13 to factorize R^c into matrices $P^c \in \mathbb{R}^{n'_D \times l}$ and $Q^c \in \mathbb{R}^{m'_D \times l}$. We change Equation 2.12 regarding factorization of R^c as follows:

$$\sum_{\text{all } C_U^d} \sum_{\text{all } C_I^{d'}} W^c_{C_U^d C_I^{d'}} \cdot \left\{ (R^c_{C_U^d C_I^{d'}} - \hat{R}^c_{C_U^d C_I^{d'}})^2 + \lambda^c . (\sum_{j=1:l} {P^c_{C_I^{d'},j}}^2 + {Q^c_{C_{U,j}^d}}^2) \right\}, \tag{4.6}$$

where ${m'}_D$ and ${n'}_D$ are the total number of clusters for items and users in domain D respectively; $R^c_{C_U^d C_I^{d'}}$ includes observed and non-observed ratings using Equation 4.3; $\hat{R}^c = R_m + P^c {Q^c}^T$ is the cluster-level predicted rating; and:

$$W^c_{C_U^d C_I^{d'}} = \begin{cases} w^c_{obs} & \text{if } R^c_{C_u C_i} \text{ is observed by Equation 4.3} \\ w^c_m & otherwise \end{cases}. \tag{4.7}$$

We execute Equations 2.14 and 2.13 step by step for Equation 4.6 to learn latent matrices P^c and Q^c. Note that we use same r_m, but different λ and w_m in the learning process of Equation 4.6. That is because of the different rate of sparsity, and also much lower dimensionality of R^c.

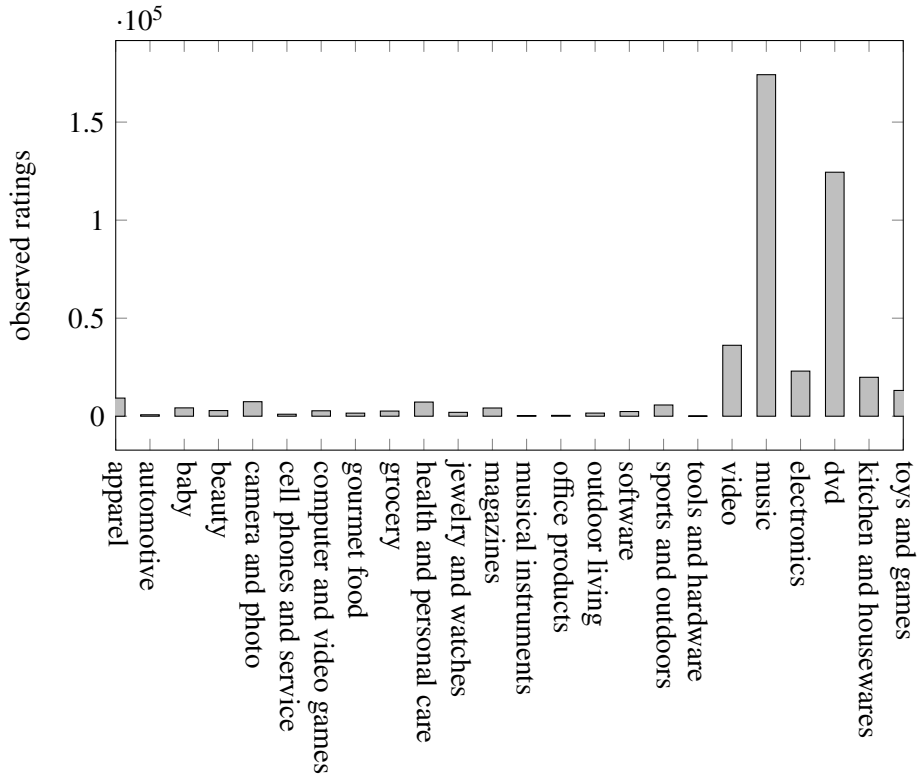


Figure 4.2: Number of observed ratings in different domains in the Amazon dataset.

4.3 Experiments

We employ two cross domain datsets in our experiments: the Amazon dataset [27] and the Epinions dataset [19,32,33]. The Amazon dataset was collected from June, 2001, to May, 2003. In total, 548,523 products were recommended, where 68% of them belongs to the domain 'books'. Thus, we ignore this domain to save our computations and also have more balanced distribution of observed ratings among domains. Figure 4.2 illustrates the number of observed ratings in the remaining domains. We select the top six domains with the largest numbers of observed ratings to employ in our cross-domain experiment. These six domains include 'DVD', 'Music', 'Video', 'Electronic', 'Kitchen and Housewares', and 'Toys and Games'. We call 'Kitchen and Housewares' as 'Kitchen', and 'Toys and Games' as 'Toys' for simplicity in the rest of this chapter. Table 4.1 presents the number of users, items, and observed ratings in each of these selected domains. As shown, the train sets are very sparse and less than 1% of ratings are observed.

Table 4.1: Number of users, items, and observed ratings in the six selected domains in the Amazon dataset.

domains	#users	#items	#ratings
electronics	18,649	3,975	23,009
kitchen	16,114	5,511	19,856
toys	9,924	3,451	13,147
dvd	49,151	14,608	124,438
music	69,409	24,159	174,180
video	11,569	5,223	36,180

Table 4.2: Percentage of user overlaps between different domains in the Amazon dataset.

domains	electronics	kitchen	toys	dvd	music
kitchen	0.051				
toys	0.028	0.041			
dvd	0.040	0.037	0.031		
music	0.032	0.028	0.020	0.119	
video	0.029	0.029	0.028	0.317	0.058

The Epinions datset [33] is extracted from Epinions[1] in June 2011. It contains reviews from users on items, trust values between users, items category, categories hierarchy, etc. This dataset contains 131,228 users, 317,755 items and 1,127,673 reviews in total. It is a very spars dataset with a 0.003 % sparsity. We employ the 10 categories with the most observed ratings of the Epinions dataset in our experiment.

The domains in the Amazon dataset only have user overlaps. Thus, there is no shared items between these domains. Table 4.2 illustrates the percentage of overlapped users between each pair of domains. As shown, 'DVD' and 'Music' domains have the most overlapped users of 31.7%, while other domains have almost 3-5% of overlaps between their user sets. We randomly split 75% of each domain for train set and dedicate the rest 25% to the test set.

In the following sections we will compare these four methods:

- **Most-Pop:** This is our basic baseline [49], [9], which uses the number of times that item *i*

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

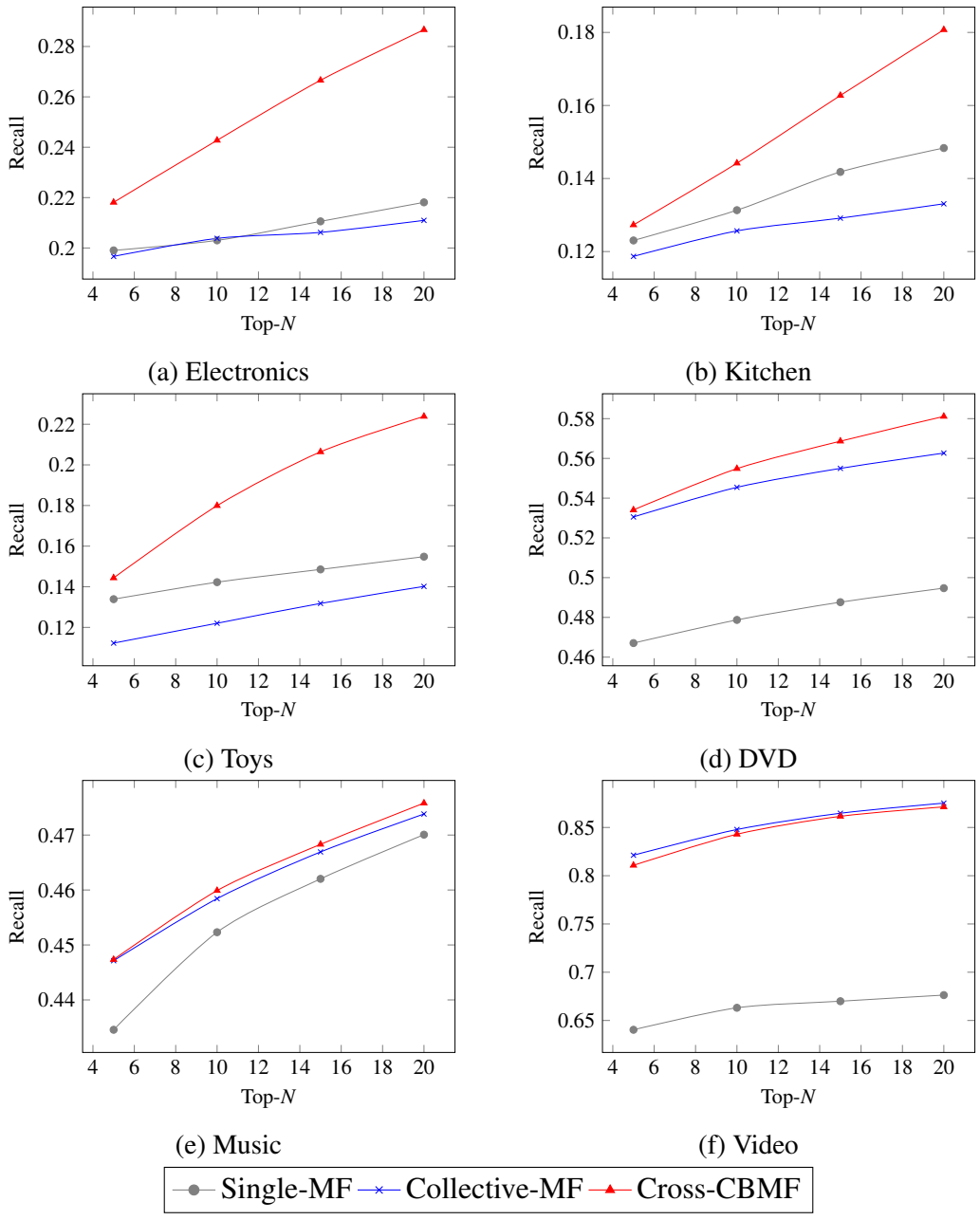


Figure 4.3: Comparing 'Single-MF', 'Collective-MF', and 'Cross-CBMF' for all users in the six selected domains in the Amazon dataset. For each domain, information of other five domains are included in the cross-domain methods.

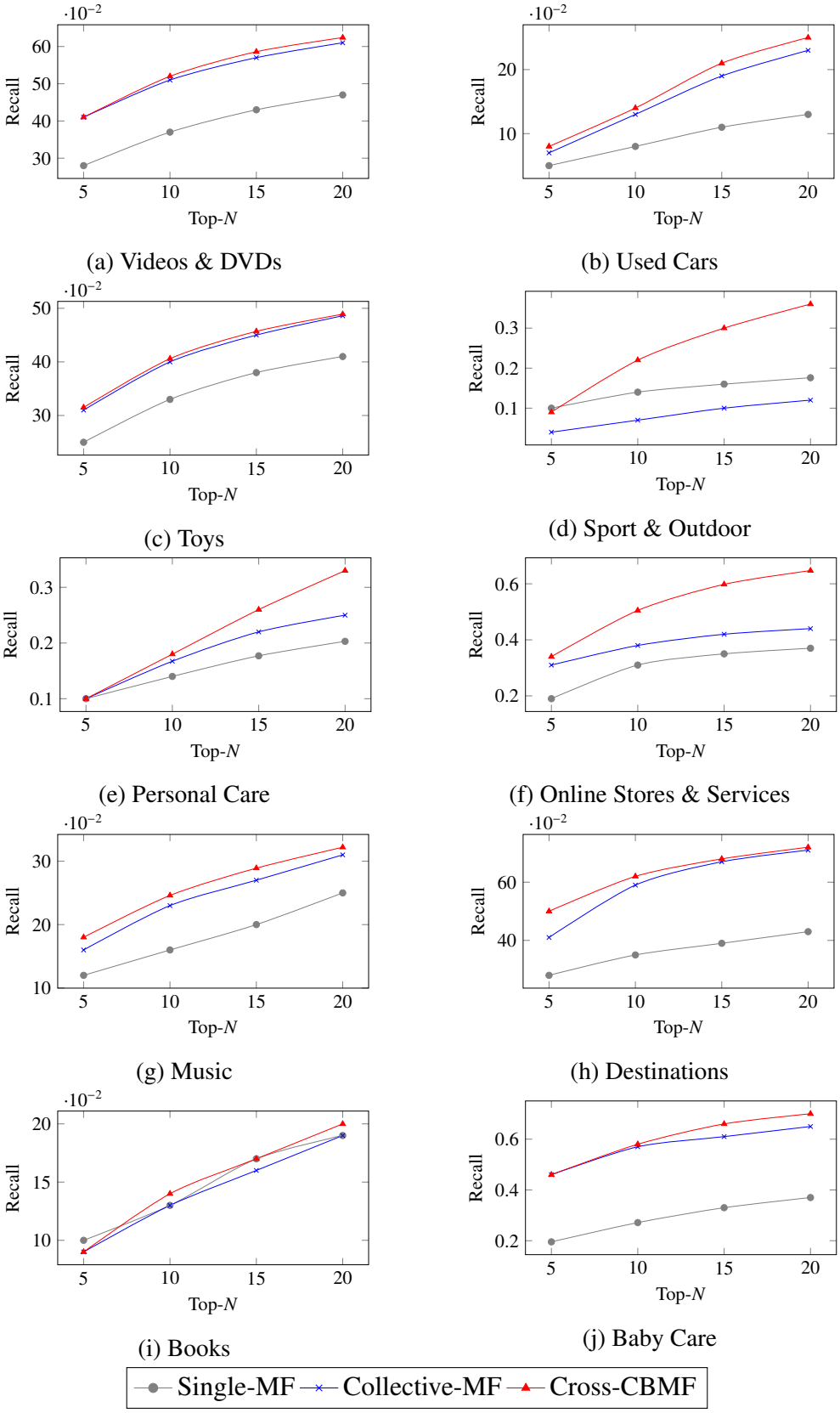


Figure 4.4: Comparing 'Single-MF', 'Collective-MF', and 'Cross-CBMF' for all users in the 10 selected domains in the Epinions dataset. For each domain, information of other nine domains are included in the cross-domain methods.

received the highest rating ($r_{ui} = 5$) for making the recommendation list. As personalized recommendations for cold-start users are inaccurate, recommending most popular items seems a reasonable option.

- **Single-MF:** This is a single-domain matrix factorization technique employing information of unobserved ratings that is proposed in [49], and is explained briefly in Section 4.2.3. In this method, we only employ the data from each domain's train set. Thus, comparing this single-domain method with our cross-domain methods will show us whether adding the extra information of auxiliary domains increases recommendation solution.

- **Collective-MF:** This is our strong baseline, which is the cross-domain extension of 'Single-MF'. Thus, to test target domain d_j we employ the train set of domain d_j adding all the ratings from the auxiliary domains. This is the traditional way of dealing with cross-domain information [18]. Hence, it does not use cluster-level recommendation. Comparing our proposed method with 'Collective-MF' shows whether our new method can utilize the data to achieve better recommendations.

- **Cross-CBMF:** This is our proposed cross-domain model, that aggregates the information of unobserved ratings from users-items level and cluster level. As described in Section 4.2.2, our proposed model employs the same information as 'Collective-MF' but utilizes it with cluster level recommendations of coarse matrix R^c to achieve more accurate recommendations.

We first compare these methods for all users (Section 4.3.1). We then limit the set of users to ones with no ratings in the train set to compare the performances of these methods for cold-start users (Section 4.3.2). As discussed earlier, we employ Top-N recommendation tasks as our evaluation metrics. Recall values are scaled in $[0, 1]$ for demonstrations. As Steck in [49] proposes, we take $w_{obs} = w^c_{obs} = 1$, and $r_m = 2$ in our experiment. Additionally, we tune our fixed parameters including α, w_m, w^c_m, λ, and λ^c via a cross-validation. We also iterate the learning process for 5 epochs to factorize both users-items level and cluster level rating

matrices. We run each experiment for 5 times with different random initializations. Thus, we report the mean result of these five runs in the following sections.

As we show in Section 3.3.1, number of clusters should not be selected too large or small. If the number of clusters is too small, the predictions of the coarse matrix will be too general, while if the number of clusters is too large, these predictions will be very close to the items-users level predictions. We use 100 clusters of items and 100 clusters of users for each domain in our experiment. Because of the large sparsity of the Epinions dataset, Applying clustering on the set of items has not achieved clusters with a good quality. Thus, we only employ the clusters of users in our final experiment in Epinions dataset

4.3.1 Performance on All Users

We include the entire six selected domains in this setup. For each domain d_j, we employ the train set of domain d_j and all the ratings from other domains to learn our cross-domain models. We then test the learned models on domain d_j's test set. Figure 4.3 presents this experiment that compares the results of three methods: 'Single-MF', 'Collective-MF', and 'Cross-CBMF' in Amazon dataset. As shown, 'Cross-CBMF' significantly outperforms the other methods in 'Electronics', 'Kitchen', and 'Toys' domains. Figure 4.4 also presents the same comparison in the Epinions dataset.

In the Amazon dataset for instance, for 'DVD' and 'Music' domains the improvements are slight, but the results of 'Cross-CBMF' and 'Collective-MF' are almost the same for 'Video' domain. It seems that employing cluster-level cross-domain information is not helpful in this specific domain. It is possibly the result of the unbalanced distribution of this domain's ratings.

Note that the results of 'Most-Pop' are removed from some figures, because this model achieves a very low recall in some setups. In 'Electronics' domain from the Amazon dataset for instance, 'Most-Pop' achieves 0.041 recall for $N = 20$.

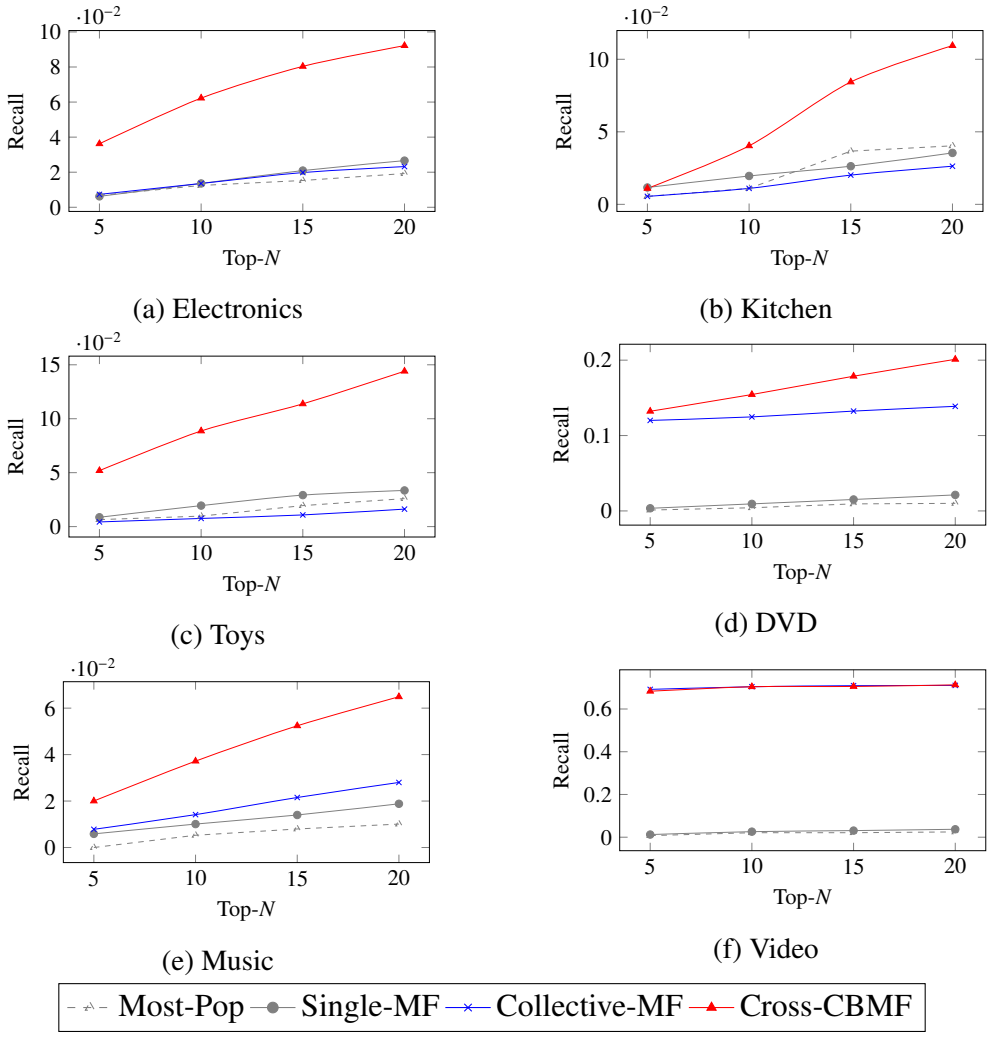


Figure 4.5: Comparing the selected methods on cold start users combining all 6 domains in Amazon dataset.

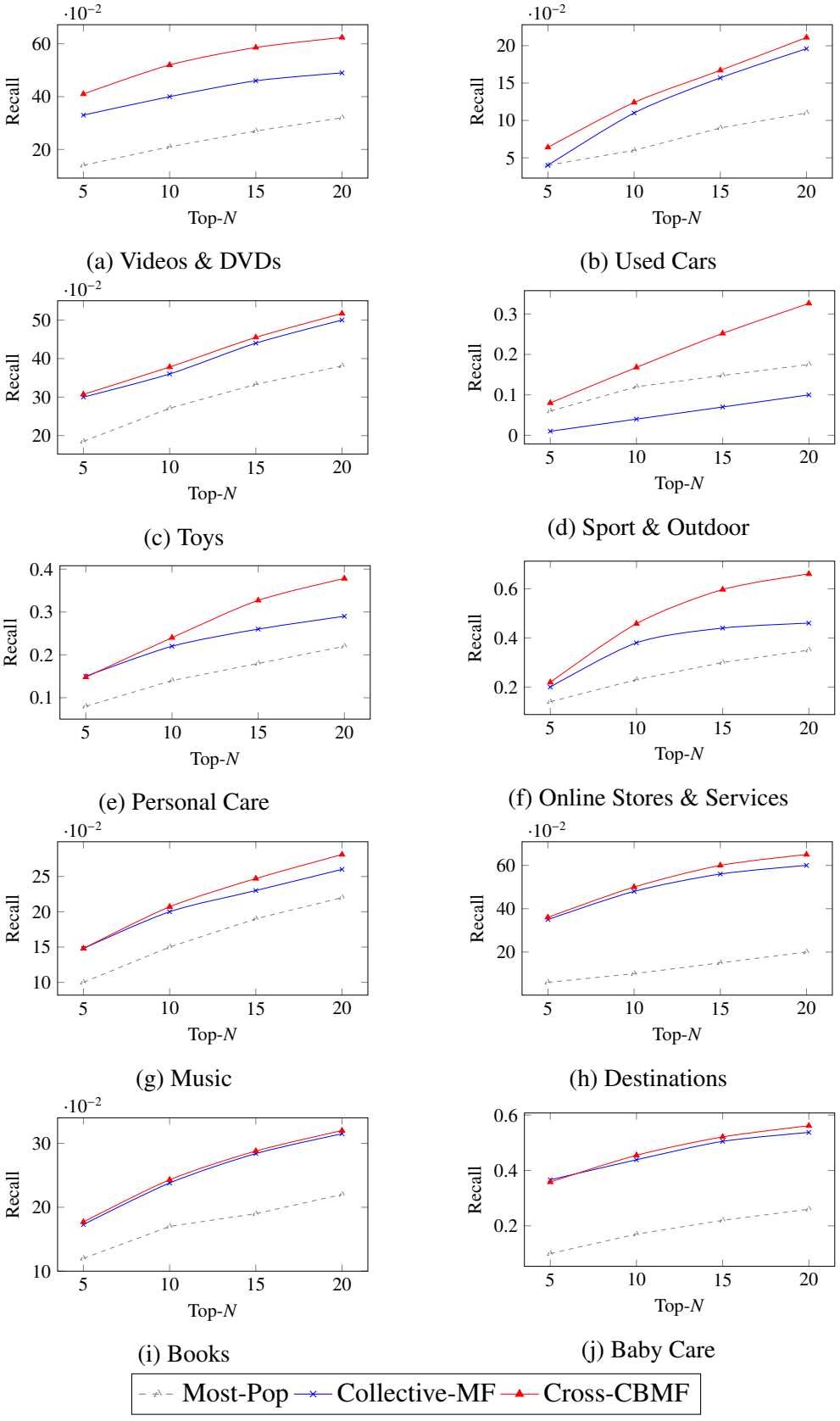


Figure 4.6: Comparing the selected methods on cold start users combining all 10 domains in the Epinions dataset.

4.3.2 Performance on Cold-Start Users

As described earlier, collaborative filtering based recommendation systems have a low performance for cold-start users. To evaluate the performance of our proposed method for cold-start users, we define cold-start users as ones who have made no ratings in the train set. Thus, single-domain methods have no collaborative information about these users. Figures 4.5 and 4.6 illustrate a comparison among selected methods. As shown, 'Cross-CBMF' dramatically increases recall for all domains except 'Video' domain (similar to Section 4.3.1) in the Amazon dataset. In the Epinions dataset, for 5 domains out of 10 selected domains the improvements are significant but slightly for the other domains.

In our experiment, we observed that the weight of unobserved ratings is much higher in the learning process of the coarse matrix than the learning process of the users-items rating matrix. In the Amazon dataset for instance, we use these fixed parameters (found via a cross-validation) in the setup that all domains are combined: $w_m = 0.0001$ and $\lambda = 0.1$ in 'Collective-MF' for all the domains, and $w_m^c = 0.9$, $\lambda^c = 0.9$ in 'Cross-CBMF' model for most of the domains. The higher values are probably due to lower dimensionality and sparsity of the coarse matrices.

Figures 4.7 and 4.8 illustrate the change of recall for 'Cross-CBMF' method by employing different values of α. As defined in Equation 4.5, for $\alpha = 0$ we do not consider the effect of cluster level recommendations and the results are similar to 'Collective-MF'. By increasing α, cluster level recommendations have more influence in the aggregated result. As shown, other than the 'Video' domain, we see improvement of recall in other domains taking appropriate values of α. In the 'Video' domain of the Amazon dataset, the value is the same for alpha value between 0 and 0.3.

To conclude, our experiments show that our proposed clustering-based matrix factorization model significantly increases the recall in top-N recommendation tasks for all users, and cold-start users in particular. For example for $N = 20$ in the Amazon dataset, our 'Cross-CBMF' method achieves a recall of 43% on average for all users compared to 39% using 'Collective-MF'. For cold-start users, our method improves recall to 21% on average, whereas including

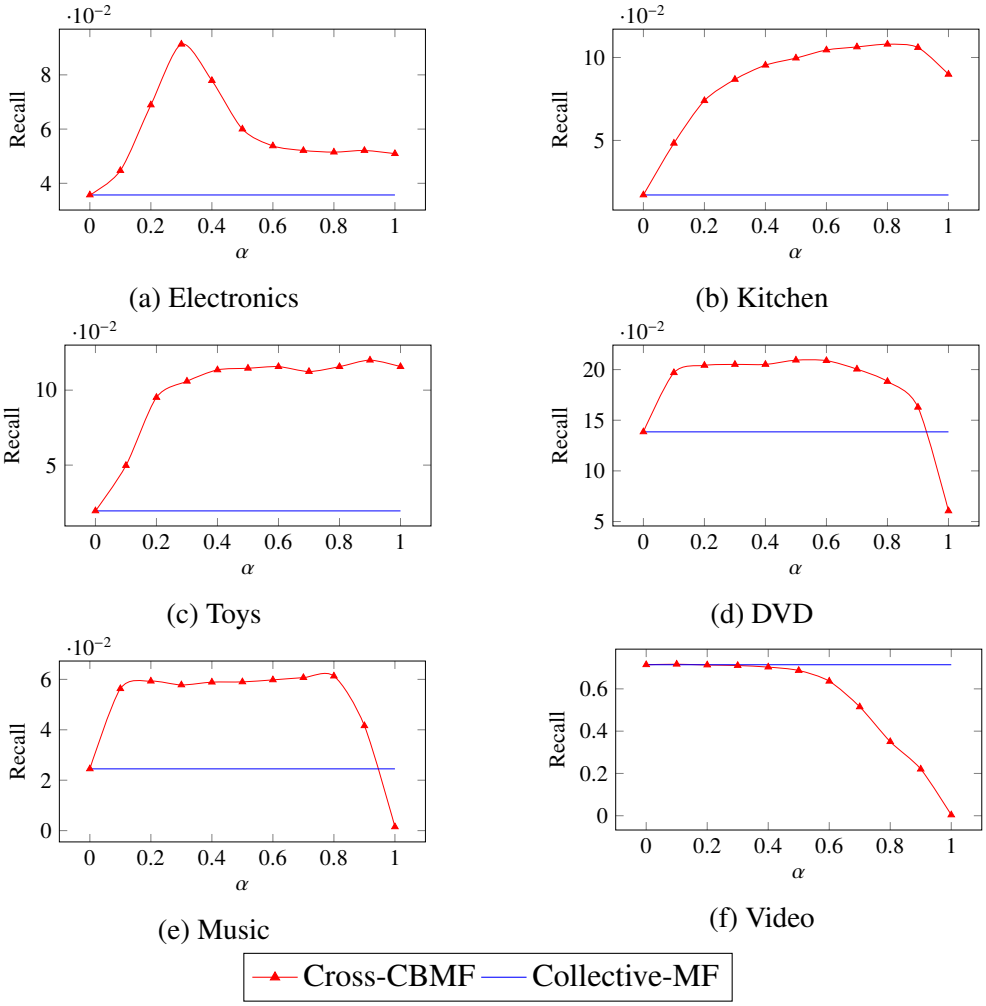


Figure 4.7: Effect of changing α value on aggregated recommendations employing top-N evaluation (N=20) in Amazon dataset. Note that for $\alpha = 0$ the recall result is same as 'Collective-MF' 's result. The effect of cluster-level recommendations increases as α increases.

data from other domains using Collective-MF' results in only 15% recall (for $N = 20$). In the Epinions dataset, our experiment shows an almost 25% total improvements of recall for cold-start users using our proposed 'Cross-CBMF'. Note that it is often difficult to make even a small improvement of recommendations especially for cold-start users. Hence, our result is quite significant.

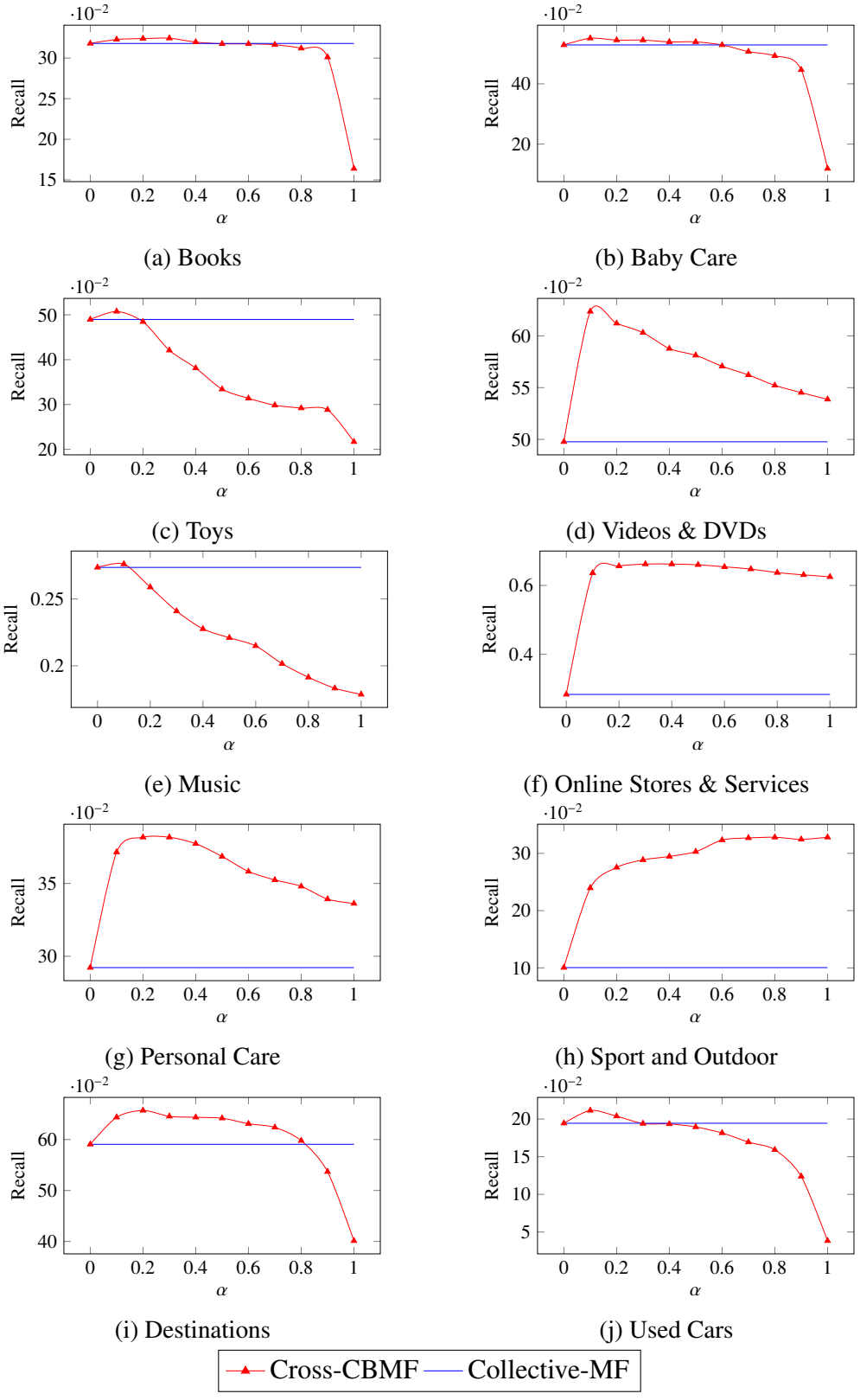


Figure 4.8: Effect of changing the α value on aggregated recommendations employing top-N evaluation (N=20) in Epinions dataset. Note that for $\alpha = 0$ the recall result is same as 'Collective-MF' 's result. The effect of cluster-level recommendations increases as α increases.

4.4 Complexity

Our proposed Cross-CBMF model has the same time and space complexity as the Alternative Least Squares (ALS) method as we once employ ALS for the traditional ratings and another time for the cluster-level ratings. Although to produce the cross-domain course matrix, a pre-processing complexity will be added to our proposed model to first factorizing the traditional rating matrix for each domain and then clustering the generated latent vectors.

4.5 Relation To Previous Work

As described earlier, employing unobserved ratings and efficient calculation of relations among the entire set of domains are two major contributions of the proposed method in this chapter. These two novelties distinguish our proposed method from current cross-domain recommendation methods. Employing unobserved ratings is shown to be dramatically effective in increasing recommendation accuracy [9, 38, 43, 49, 50]. Consequently, we expect that our proposed method will outperform the current cross-domain methods on Top-N recommendation tasks as they are learned only based on observed ratings and with respect to improving prediction accuracy. This is because (as Cremonesi shows in [9]) methods with a good prediction accuracy do not always results in good recommendations accuracy. Moreover, many well-observed cross-domain methods such as the proposed models in [8], [51], [29] require a heavy computation to find possible relations between each two domains. However, as we described in Section 4.2.1, we find these relations among the entire set of domains in one efficient step.

Li et al. in [29] propose a similar cluster-level integration of ratings in a cross-domain recommendation system. They adopt the orthogonal non-negative matrix tri-factorization algorithm to construct a cluster-level rating matrix, called ‘Codebook’. Our proposed coarse matrix (Equation 4.1) is similar to this Codebook with two differences. First, the Codebook does not capture unobserved rating values among cluster of users and items. Second, these

Codebooks are domain-specific. Thus, for any two domains they have to apply an expensive transferring algorithm , called 'Codebook Transfer', to find possible relations between pairs of Codebooks. However, our proposed method employ the extra information of unobserved ratings in the clusters level and users-items level. Our method also finds the relations among the coarse matrices in one step. Gao et al. [13] propose an almost similar method based on Li's proposed Codebook. They consider similar explicit cluster-level latent space for users from different domains while the items in each domain may hold their domain-specific latent vectors. Moreno et al. in [37] also generalize Li's method to transfer the auxiliary knowledge from multiple domains into one domain in contrast with Li's model which is based on transferring knowledge from one domain to another. However, both methods suffer from the same limitations as the Li's method.

Hu et al. in [18] integrates information from unobserved ratings into a cross-domain triadic factorization model. They merge domain-specific items-users rating matrices into a cubic users-items-domains rating matrix. Their proposed tri-factorization model is then applied to factorize this cubic rating matrix into users, items, and domains latent space. They show empirically that their method outperforms unobserved-ratings integrated matrix factorization (same as the model that we call 'Collective-MF' in our experiment). However, cold-start users are ignored in their experiments, where they run their experiment over users with at least 30 observed ratings. Moreover, instead of finding users-items relations among different domains, we consider relations between the cluster of users and items among different domains. We show in Section 4.3 that our proposed method make a significant improvement of top-N recommendation task for cold-start users.

Other related papers can be categorized into cross-domain transfer learning for recommendation and cross-domain collaborative filtering. Yin Zhu et al. in [58] propose a Heterogeneous Transfer Learning for Image Classification. Their proposed method employs Matrix Factorization to transfer useful knowledge in texts into image classification model. Although they do not address the recommendation problem directly, but their interesting model can be com-

bined with our proposed model to make a context-aware recommendation system. Jiang et al. in [21] propose a novel Hybrid Random Walk (HRW) to integrate multiple heterogeneous domains such as users' social networks to improve the recommendation accuracy. Random walk based models in general provide simple solutions to integrate knowledge of different domains. However, they are sensitive to the number of steps in their random walk models, and their complexity dramatically increases by growing the number of steps. The authors in [30] propose a Gaussian Probabilistic Latent Semantic Analysis (GPLSA) model that consider the consistency between the knowledge in two domains and only transfer the consistence auxiliary information between cross domains. Their proposed selective transfer learning transfer the knowledge in user-item level which can be compared with Hu et al.'s proposed model in [18]. However, we generalize the user-item matrix to reduce the sparsity. We then transfer the knowledge in two levels: clusters level and users-items level.

In [6], Chen et al. employ the same idea of clustering items and users in latent space and transferring the clustering level knowledge between domains. However, in their proposed method they need to learn a transition model between the clustering representation of each two domains, which is expensive to be generalized for multiple domains. This methods may be compared to Li's Codebooks [29]. They also employ the auxiliary information of items' contents and also users' social network into their cross domain recommendation model. Again, our proposed model can easily handle including the auxiliary information from multiple domains without a need for learning a domain-domain mapping function.

In addition, Shi et al. in [48] and Li in [28] provide a complete survey on cross-domain recommendation systems. Pan et al. in [39] present a complete survey on Transfer Learning in general for transferring knowledge between multiple domains in machine learning. Adomavicius in [1] also provides a good survey about state-of-the-art methods in recommendation system.

Social networks are also important source of knowledge which can be included as auxiliary domains into cross-domain recommendation models. Jamali et al. in [19] provide a promising

model to include the social trusts to improve the recommendation accuracy. As mentioned earlier, Chen et al. in [6] include the social information beside other auxiliary information for this purpose. Our proposed method may also be applied in social networks to include cluster-level information from social/trust networks for building a more accurate cross-domain personalization model.

Chapter 5

Clustering-Based Personalization In Adaptive Webs

5.1 Introduction

Content Marketing is any marketing that involves the creation and presentation of media and publishing content in web to acquire and attract users. This content can be presented in a variety of formats, including images, videos, texts, etc. In 2014, 93% of B2B marketers employ content marketing. Moreover, the conversion rate is nearly 6 times higher for content marketing adopters than non-adopters (2.9% vs 0.5%) [40]. That is the reason that WWW rapidly switches from static web to adaptive web where businesses can make different versions of content to target more users with diverse preferences. Adaptive web is a rare case of personalization with millions of cold-start user and only few versions of contents, where both users and items are unstable and change frequently. These two challenges beside speed and comprehensibility of personalization (Section 5.2) are the main reasons that current personalization methods are less useful in this scenario.

Moreover, the personalization task in traditional adaptive web relies on manually dividing users based on their locations and available profiles. Agencies then employ A/B testing of

different contents on different segment of users to find the best matches. A new user then will be mapped to one of these predefined segments and will be presented by this segment's fitted version. However, this rigid rule-based approach is costly and cannot comprehensively include all users with diverse preferences.

In this chapter, we employ advance clustering techniques beside deep learning to automate this personalization task. This is part of our smart adaptive web platform, called Morphio, which is designed to target more users with smart personalized contents. In Morphio, we include external databases to generate an extended profile of users including locations, income average, etc. We then employ our proposed clustering technique in [34, 35] to cluster these users based on their generated profiles and also their sparse set of page visits. Finally, for a new user, instead of matching her to only one related cluster, we find a distribution of cluster assignments such as $P(cluster_i|user_j)$ employing their profile information. We then employ this distribution of soft assignments beside the successful versions of each cluster to find her matched version. Users have diverse preferences which cannot be easily captured with rigid rule-based systems. This soft assignment of users to the generated clusters empowers our personalization system to cover more users with variety of interests. Clustering users and training the classifier regarding the clusters and not the versions gives our personalization model extra scalability and comprehensibility; First, we do not have to train a classifier for each page with multiple versions. Second, we do not have to re-train our classifiers when we update the versions. Third, agencies are able to supervise the matched version of content to each cluster of users. In addition, in *Content Analytic Module* in Section 5.4, we also employ our proposed clustering-based recommendation system to suggest smart contents for current web pages. This module allows agencies to improve their produced content using our data-driven insights. Our ongoing real time experiment shows a significant improvement of user conversion employing our proposed clustering-based personalization.

5.2 Morphio Platform

Morphio is a research and development project in collaboration with Arcane inc.[1], an emerging Canadian digital marketing agency. Arcane's content marketing system was based on individual expertise, traditional A/B testing, and several possible tries and errors. Morphio has been designed as a smart adaptive web platform with two main goals; First, to suggest this agency goal-driven smart contents. And second, to target more users with personalized contents. Figure 5.1 illustrates an overview of Morphio's platform. Morphio mainly works based on IP targeting and contains the following 4 major modules (in this chapter we focus on our proposed personalization module only):

- *Personalization Module:* to target users with diverse preferences by different versions of contents (Section 5.3).
- *Content Analysis Module:* to analyse the current contents considering their positions in the webpages and their impact to increase the conversion rate. In addition, to suggest new contents to content marketing agencies to improve their products.
- *Page Analysis Module:* to classify weak and strong pages based on their impact in achieving predefined goals.
- *User Analysis Module:* to classify clusters of users whom are presented by weak or strong contents.

A light JavaScript based software lets content marketing agencies to connect their websites to our Java implemented web servers. We then track user activities based on HTTP requests to store their page visits and click events. Agencies can define one of multiple conversion goals for each of their registered websites. As mentioned earlier, adaptive web is a rare scenario in recommendation systems. The followings are four challenges that our proposed personalization module has to address:

[1] www.arcane.ws

- More than 90% of users are cold-start users.

- Items and users are both unstable. We identify each user based on her IP and session ID which both are unstable and can be changed from time to time. Contents are also frequently replaced by new contents due to different advertising strategies.

- Speed. It is one of the main challenges in this personalization task. Recent study [40] shows that even 1 second late of page loading will significantly reduce the user conversion rate. Thus, once a new user request a page until we decide which version of content should be presented to her, it should not take more than split of a second.

- Comprehensibility. Agencies tend to understand which versions of content suit which group of people better. Thus, personalization models which act as black boxes are not good candidates in this context.

In the next section we will explain how our proposed method address these challenges.

Figure 5.1: A general view on our designed personalization platform.

5.3 Personalization Module

We propose a scalable personalization model as the core of our *Morphio* platform. In Morphio, we identify each user by her IP and session ID. We then employ the available information in the HTTP request to generate a profile for each user. Each HTTP request contains user's IP, session ID, the device that she is using (Mobile or PC), operating system, and the browser that she is working with. Usual visiting days and hours also can be easily obtained from user activities. By employing IP, we also can estimate user's location and consequently her city, province, and country. We then include auxiliary databases to enrich user profiles by extra knowledge about these locations that users live in such as income average.

As mentioned earlier, both users and items are unstable and change frequently. Also, in real world scenario we do not have users with many page visits. Users usually visits 1-2 pages and then decide to convert or leave the website. Because of these obstacles we prefer to train our personalization model based on new users only and do not waste any resources on our old users. It explains why we need a pure content filtering method to personalize our unstable content in only split of a second. Thus, K-Nearest Neighbor (KNN) or other traditional machine learning model such as SVM and deep learning seems good choices for our personalization model. However, as mentioned in Section 5.2, agencies need a comprehensible model to understand which version of content works for which group of people. Moreover, contents are unstable too. Thus, we cannot afford to train the personalization model every time we change a piece of content. In this chapter we present our clustering-based personalization model which address these indicated challenges.

We first employ the generated profiles beside the available page visits to cluster users into several clusters. As there are millions of users with hundreds or maybe thousands of pages we need a scalable clustering method which can be applied regularly on the set of users. Thus, we employ our proposed scalable clustering technique in [34, 35] that is based on reducing the feature space using matrix factorization [26]. We first produce the following users-pages matrix:

$$R = \begin{array}{c} \\ u_1 \\ u_2 \\ \vdots \\ u_n \end{array} \begin{array}{c} \begin{array}{cccc} p_1 & p_2 & \dots & p_m \end{array} \\ \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1m} \\ r_{21} & r_{22} & \dots & r_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n1} & r_{n2} & \dots & r_{nm} \end{pmatrix} \end{array}$$

where n is the number of users, m is the number of pages, and r_{nm} is 1 if user n has visited page m and 0 otherwise.

Matrix factorization generalizes user preferences and reduce the sparsity of this users-pages

matrix R. It is based on factorizing rating matrix R into latent matrices P and Q in a lower space, where:

$$R = P.Q^T$$

We employ the stochastic gradient descent [26] to learn latent matrices P and Q. Let's assume d is the length of the latent vectors in these two matrices. Thus for each user j we have following latent vector P_j:

$$P_j = [p_{1j}, p_{2j}, \ldots, p_{dj}],$$

and for page i we have following latent vector Q_i:

$$Q_i = [q_{1i}, q_{2i}, \ldots, q_{di}].$$

We then add user profiles as new columns to the latent vector P and apply K-Means clustering on this new matrix to produce clusters of users. Let's assume that C is the set of these found clusters $C = \{C_1, C_2, .., C_{n'}\}$. For each page p that contains more than one version, we first do a random A/B testing for a predefined period of time to find the most successful version of that page for each cluster of users based on a defined goal. These clusters gives our personalization model a good level of comprehensibility. Thus, agencies can easily understand the way that we use the available contents. In addition, we do not have to re-train our personalization model by every change in the contents.

We employ user profiles to classify users into the found clusters in C. Thus, for each cluster C_i we gather the positive and negative instances and employ *deep learning* to classify new user. Our multi-classes classifier will be trained on user profiles to predict the clusters that new user may belong to. Hence, we can predict a distribution over the clusters as the classes such as $P(C_i|u_j)$ for cluster i and user j. Let's assume new user j visits multi-version page p with versions v_1 and v_2. Our final predicted version will be the one which maximize the following

function:

$$\underset{s}{argmax} \sum_{i} P(C_i|u_j).P(v_s|C_i)$$

where $P(v_s|C_i)$ is the success chance of version v_s for the users in cluster C_i. $P(v_s|C_i)$ is calculated during the A/B testing process. For the rest of this chapter, we call this method as *Clustering-Based Personalization (CBP)*.

In Section 5.5, we compare our proposed solution with three models. First, we compare CBP with random version assignment which is simply selecting a random version for each new user. Second, in the *Winner* model we present the most popular version to users. Third, we compare CBP with traditional *K-Nearest Neighbor (KNN)*. In this method we employ users' profiles and page visits to find the K most similar users to new user u_j and then present her the version, which maximizes the following function:

$$\underset{s}{argmax} \sum_{t} P(v_s|u_t)$$

where $P(v_s|u_t)$ is the success chance of version v_s for user u_j.

5.4 Content Analytic Module

Morphio's content analytic module consists of splitting each web page of contents into its elements and analysing the impact of each of these pieces of contents. Thus, for each web page of a registered website we extract following elements: texts, keywords, images, background colors, text colors, font sizes, images main colors, font family, etc. We also think the position of that element is also an important factor in a page. Thus, we store these elements beside the frame of the page that they have been presented in. We consider 4 different frames including left-top, left-down, right-top, and right-down. Hence, each elements will be stored based on its position in these frames. We call each of these positioned element as an attribute, which

left-top	right-top
left-down	right-down

Table 5.1: Each web page will be cut into different splits using these frames.

contains a frame ID and a page element. Table 5.1 illustrates the frames that we use to store the position of each elements.

Our content analytic module then employs a ranking algorithm beside the page visits to suggest new content for each page. These suggested contents are meant to improve these pages' impact. To achieve such a recommendation model, we employ the users-pages matrix V, where:

$$V_{up} = \begin{cases} 5 & \textit{if user } u \textit{ has visited page } p \\ & \textit{and has achieved a goal} \\ 3 & \textit{if user } u \textit{ has visited page } p \\ & \textit{but it has not achieved any goal} \\ 1 & \textit{otherwise.} \end{cases} \tag{5.1}$$

We also define attributes-pages matrix E as follows:

$$E_{ep} = \begin{cases} 1 & \text{if attribute } e \text{ is observed in page } p \\ 0 & \textit{otherwise.} \end{cases} . \tag{5.2}$$

We then recalculated matrix E to apply the visits' feedback on the attributes as follows:

$$E^* = (E \times V^T) \times V$$

Now, E^*_{ep} represents impact of attribute e in page p on targeting visitors and achieving

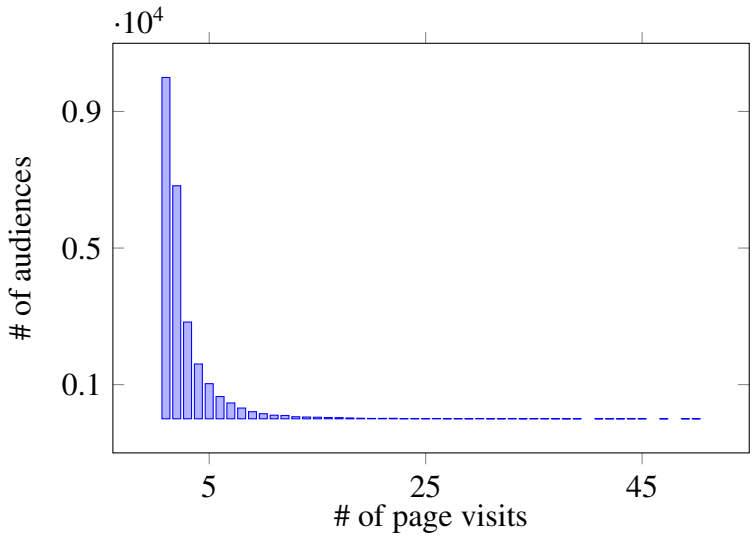


Figure 5.2: The distribution of audiences versus their number of page visits.

more user conversions. Thus, by factorizing E^*_{ep}, we can rank the most effective attributes to be employed in these pages and increase their impact to increase the user conversion. We apply our proposed clustering-based matrix factorization in Section 4.2.3 on E^*_{ep} to predict and rank the unknown values in this new generated matrix. This module allows agencies to improve their produced content using our data-driven suggestions and insights.

5.5 Experiment

As mentioned earlier, this work is part of our *Morphio* platform in contribution with Arcane inc. Our personalization module is still in an early stage. We could connect only 2 websites from Arcane's clients to this platform as testbeds. From those two registered websites, we only had the chance to have 2 versions of landing page from one of these sites to test and evaluate our proposed personalization method. However, we still had this opportunity to evaluate these methods in a real time scenario with actual users. These two designed versions were visually different and each was emphasising on one of this website's line of product. The goal of A/B testing has been set to optimize the user conversion which has been defined based on submitting a contact form to request an appointment.

We run our designed experiment on these two versions for 3 months, with 86 conversions and approximately 7000 impressions in total. Figure 5.2 illustrates the distribution of page visits for users in a 4 months period of time. As shown, over 40% of users are new and 90% of users has 4 or fewer page visits. These results prove the rare case of our novel personalization scenario. When an user have visited the under testing landing page, we identify her based on her IP and session ID. If she was an old user with a conversion we present her the version that she has been presented before. If she was a new user, we randomly employ one of those 4 designed methods in Section 5.3 to choose her related version of content.

Figure 5.3 illustrates a comparison between these four methods based on their conversion rate. Note that 1-2% conversion rate may looks small but it is natural in this context. As shown, our proposed CBP method significantly outperforms all other methods in practice by 1.98% conversion rate. After CBP, KNN achieves the next best result by 1.53% conversion rate, while our CBP method is much faster than KNN in the prediction time. All the parameters including K in KNN and the *deep learning*'s related parameters have been selected in a separate off-line experiment.

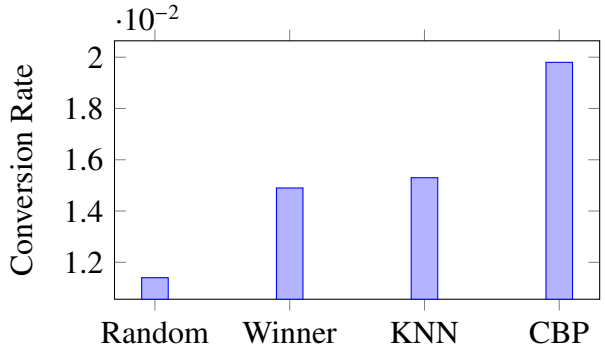


Figure 5.3: Comparing our proposed CBP method with three other methods regarding user conversion optimization.

5.6 Relation to Previous Work

Mobasher in [36] and Bunt in [5] present a comprehensive introduction on adaptive web and employing data mining to personalize WWW. The general idea of using user's location to make the recommendations is highly related to location-aware recommendations. Location-aware models beside other context-aware recommendation systems have been well studied in the last decade. For instance, Rendle et al. in [44] propose a Tensor Factorization technique to factorize cubic rating matrix users-items-contents. Wetzker et al. in [53] propose a hybrid solution by employing PLSA on a merged representation of user-item-tag observations. Adomavicius et al. in [2] employ ratings aggregation to reduce the multi-dimensional (contexts-users-items dimensions) rating matrix to the traditional 2 dimensional rating matrix. Hariri et al. in [16] propose a KNN technique and employs inferred topics (context) to calculate the item-item similarity.

Chapter 6

Summary, and Conclusions

Personalization, or in a more general term recommendation system, is a key part of the information and e-commerce ecosystem [11]. Businesses offer millions of items through different channels to target people around the world. Large number of items is an obstacle for users who try to find the items that they are looking for. Recommendation systems address this problem by providing powerful methods which enable users to filter through these large repositories based on their preferences. Thus, diverse preferences of users beside the emerging growth of contents force businesses to widely employ personalization technologies. Recent studies show a significant improvement of revenue and user conversion rate for personalization adopters. Accuracy, scalability, data sparsity and comprehensibility are main challenges to design practical recommendation systems.

In this thesis, we employ clustering to include neighborhood information in recommendation systems. We first cluster users and items separately. Then, we present the average preferences of the users in each user clusters on the items in the item clusters in a new cluster-level rating matrix. This coarse matrix generalizes the observed interests to reduce the sparsity of the original rating matrix. By including our proposed cluster-level rating matrix, we try to improve the recommendation accuracy of the-state-of-the-art recommendation systems, while preserving their scalability.

In Chapter 3, we employ our proposed cluster-level rating matrix to improve a number of well-studied recommendation models in single domains. We fist propose an scalable clustering technique to cluster items and users, and producing the cluster-level rating matrix. A number of matrix factorization methods are then applied to this coarse matrix to predict the future cluster-level interests. We then aggregate the traditional user-item rating predictions with our cluster-level rating predictions to improve the personalization accuracy further. We employ RMSE evaluation metric in our experiment on two well-known datasets: Netflix and Movielens. Our extensive experimental results show that our new approach, when applied to a variety of existing matrix factorization methods, improves their rating prediction accuracy.

We also employ four different clustering methods to compare the impact of clustering quality on our proposed model's rating prediction accuracy. Our experiment in the both datasets shows that improving the quality of clusters increases rating prediction accuracy of our proposed clustering-based matrix factorization (CBMF) model. In Section 3.3.4 accuracy is improved even further by employing clusters in variety of sizes in an extension of our CBMF model.

These extension models have almost the same complexity as the non-extended models. However, they add a complexity for the preprocess in the clustering step. For instance, the training time of the extensions are less than twice of the non-extended models in the MovieLens100k dataset. The clustering was also not considerably time consuming because we perform the clustering on the low dimension latent vectors. For instance, the clustering of the Netflix's users takes less than a hour using the Rapidminer[1] software in our PC with 3.30 GHz CPU. Thus, the extended models keep the scalability of those models.

In Chapter 4, we extend our proposed clustering-based recommendation system by utilizing data from auxiliary domains to achieve better recommendations, especially for cold-start users. Traditional recommendation systems assume that items belong to a single domain. However, at the present time users rate items or provide feedback in different domains. Thus, businesses

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

intend to empower their business intelligence by this cross-domain information to generate better recommendations and consequently improve their revenue. Most previous works in cross-domain recommendations ignore a significant part of available information, unobserved ratings. These methods also mainly focus on improving prediction accuracy, often known in terms of RMSE, which has been criticized over the last few years. We extend our previous work on clustering-based matrix factorization in single domains into cross-domains by utilizing recent results on considering unobserved ratings as negative feedback. We define a cross-domain 'coarse' matrix, which captures the shared preferences between clusters of users and cluster of items in same or different domains. Using this coarse matrix, we propagate the observed ratings into the cluster level unobserved ratings to reduce the sparsity of traditional rating matrices. Finally, our proposed clustering-based matrix factorization aggregates the recommendations from these two levels, and it effectively utilizes cross-domain data to improve recommendation accuracy. Our experiments show that our method improves recommendation accuracy for all user and cold-start users in particular. For instance, our method achieves a recall of 43% on average for all users compared to 39% using the previous methods. For cold-start users, our method improves recall to 21% on average, whereas those previous methods result in 15% recall. We also observe almost 25% improvement of recall in the Epinions dataset. It is often difficult to make even a small improvement in recommendations, and especially for cold-start users. Thus, our result is quite significant.

Finally, in Chapter 5, we review our contribution in a smart adaptive web platform called *Morphio*, which is designed to help content marketing agencies to produce smart contents and target more audiences. *Morphio* is a research and development project collaborated with Arcane Inc., an emerging local digital marketing company. *Morphio*'s personalization module applies our clustering-based approach on the content information beside collaborative resources to propose a hybrid personalization system. This solution is based on clustering the old users and find the best version of content for each of these generated clusters. Our proposed method then soft assigns new users into these clusters employing their profiles to conclude their fit-

ted version of content. Our ongoing real time experiment shows a significant improvement of user conversion employing our proposed clustering-based personalization comparing to the traditional A/B testing models. Our proposed model improve the conversion rate by almost 32%.

As the future works, our proposed clustering-based matrix factorization model can be extended to perform both the clustering and the training phase simultaneously. At the present time, we have two separate steps in our proposed training process. We first clustering users and items and then employ the found clusters in the training process. However, finding these clusters and also finding the optimum numbers for the clusters during the training process have several advantages including a less complexity for our CBMF model.

To extend Morphio project, we are going to employ our proposed CBMF model to generate an optimized version of content for each cluster of users. In this way we can select pieces of content from each defined versions and produce an optimized combination of them which suites a given user's preferences. Also, we currently produce content suggestions based on a given page. A useful research path to continue would be suggesting content based on a given content. Thus, in this way we can get a list of suggested content such as colors based on our queried content which can be a color or any other pieces of content.

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

[2] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1):103–145, January 2005.

[3] Alex Beutel, Kenton Murray, Christos Faloutsos, and Alexander J. Smola. Cobafi: Collaborative bayesian filtering. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 97–108, New York, NY, USA, 2014. ACM.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[5] Andrea Bunt, Giuseppe Carenini, and Cristina Conati. Adaptive content presentation for the web. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 409–432. Springer Berlin Heidelberg, 2007.

[6] Wei Chen, Wynne Hsu, and Mong Li Lee. Making recommendations from multiple domains. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 892–900, New York, NY, USA, 2013. ACM.

[7] Mark Connor and Jon Herlocker. Clustering items for collaborative filtering. In *In Proceedings of the ACM SIGIR Workshop on Recommender Systems*, Berkeley, CA, 1999.

[8] P. Cremonesi, A. Tripodi, and R. Turrin. Cross-domain recommender systems. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 496–503, 2011.

[9] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM.

[10] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 107–144. Springer US, 2011.

[11] Michael D. Ekstrand, John T. Riedl, and Joseph A. Konstan. Collaborative filtering recommender systems. *Found. Trends Hum.-Comput. Interact.*, 4(2):81–173, February 2011.

[12] Ignacio Fernández-Tobías, Iván Cantador, Marius Kaminskas, and Francesco Ricci. Cross-domain recommender systems: A survey of the state of the art. In *Spanish Conference on Information Retrieval*, 2012.

[13] Sheng Gao, Hao Luo, Da Chen, Shantao Li, Patrick Gallinari, and Jun Guo. Cross-domain recommendation via cluster-level latent factor model. In Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Åelezn, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 8189 of *Lecture Notes in Computer Science*, pages 161–176. Springer Berlin Heidelberg, 2013.

[14] Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *Proceedings of the Fifth IEEE International Conference on Data*

Mining, ICDM '05, pages 625–628, Washington, DC, USA, 2005. IEEE Computer Society.

[15] Modou Gueye, Talel Abdessalem, and Hubert Naacke. A cluster-based matrix-factorization for online integration of new ratings. In *Journées de Bases de Données Avancées (BDA)*, pages 1–18, 2011.

[16] Negar Hariri, Bamshad Mobasher, and Robin Burke. Context-aware music recommendation based on latenttopic sequential patterns. In *Proceedings of RecSys '12*, pages 131–138, New York, NY, USA, 2012. ACM.

[17] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 230–237, New York, NY, USA, 1999. ACM.

[18] Liang Hu, Jian Cao, Guandong Xu, Longbing Cao, Zhiping Gu, and Can Zhu. Personalized recommendation via cross-domain triadic factorization. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 595–606, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[19] Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 135–142, New York, NY, USA, 2010. ACM.

[20] Mohsen Jamali, Tianle Huang, and Martin Ester. A generalized stochastic block model for recommendation in social rating networks. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 53–60, New York, NY, USA, 2011. ACM.

[21] Meng Jiang, Peng Cui, Fei Wang, Qiang Yang, Wenwu Zhu, and Shiqiang Yang. Social recommendation across multiple relational domains. In *Proceedings of the 21st ACM*

International Conference on Information and Knowledge Management, CIKM '12, pages 1422–1431, New York, NY, USA, 2012. ACM.

[22] J. A. Konstan and J. T. Riedl. Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction*, 22:101–123, 2012.

[23] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM.

[24] Yehuda Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Trans. Knowl. Discov. Data*, 4(1):1:1–1:24, January 2010.

[25] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 145–186. Springer US, 2011.

[26] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[27] Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. *ACM Trans. Web*, 1(1), May 2007.

[28] Bin Li. Cross-domain collaborative filtering: A brief survey. In *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*, pages 1085–1086, 2011.

[29] Bin Li, Qiang Yang, and Xiangyang Xue. Can movies and books collaborate?: Cross-domain collaborative filtering for sparsity reduction. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 2052–2057, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[30] Zhongqi Lu, ErHeng Zhong, Lili Zhao, Evan Wei Xiang, Weike Pan, and Qiang Yang 0001. Selective transfer learning for cross domain recommendation. *CoRR*, abs/1210.7056, 2012.

[31] Benjamin M. Marlin and Richard S. Zemel. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 5–12, New York, NY, USA, 2009. ACM.

[32] Paolo Massa and Paolo Avesani. Trust-aware bootstrapping of recommender systems. In *ECAI 2006 Workshop on Recommender Systems*, pages 29–33, 2006.

[33] Simon Meyffret, Emmanuel Guillot, Lionel Mdini, and Frdrique Laforest. RED: a Rich Epinions Dataset for Recommender Systems. Technical Report RR-LIRIS-2012-014, LIRIS UMR 5205 CNRS/INSA de Lyon/Universit Claude Bernard Lyon 1/Universit Lumire Lyon 2/cole Centrale de Lyon, October 2012.

[34] Nima Mirbakhsh and Charles X. Ling. Clustering-based factorized collaborative filtering. In *Proceedings of the 7th ACM conference on Recommender systems*, RecSys '13, pages 315–318, New York, NY, USA, 2013. ACM.

[35] Nima Mirbakhsh and Charles X. Ling. Improving top-n recommendation for cold-start users via cross-domain information (accepted to publish). *the Transactions on Knowledge Discovery from Data (TKDD)*, 2015.

[36] Bamshad Mobasher. Data mining for web personalization. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 90–135. Springer Berlin Heidelberg, 2007.

[37] Orly Moreno, Bracha Shapira, Lior Rokach, and Guy Shani. Talmud: Transfer learning for multiple domains. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 425–434, New York, NY, USA, 2012. ACM.

[38] Xia Ning and George Karypis. Sparse linear methods with side information for top-n recommendations. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 155–162, New York, NY, USA, 2012. ACM.

[39] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, Oct 2010.

[40] Joe Pulizzi and Ann Handley. B2b content marketing. *benchmarks, budgets, and Trendsnorth America*, 2014. Available at http://www.iab.net/media/file/B2BResearch2014.pdf.

[41] Anand Rajaraman and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA, 2011.

[42] Al Mamunur Rashid, George Karypis, and John Riedl. Learning preferences of new users in recommender systems: an information theoretic approach. *SIGKDD Explor. Newsl.*, 10(2):90–100, December 2008.

[43] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press.

[44] Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of WSDM '10*, pages 81–90, New York, NY, USA, 2010. ACM.

[45] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[46] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 253–260, New York, NY, USA, 2002. ACM.

[47] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

[48] Yue Shi, Martha Larson, and Alan Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Comput. Surv.*, 47(1):3:1–3:45, May 2014.

[49] Harald Steck. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 713–722, New York, NY, USA, 2010. ACM.

[50] Harald Steck. Evaluation of recommendations: Rating-prediction and ranking. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 213–220, New York, NY, USA, 2013. ACM.

[51] Jie Tang, Sen Wu, Jimeng Sun, and Hang Su. Cross-domain collaboration recommendation. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 1285–1293, New York, NY, USA, 2012. ACM.

[52] Andreas Töscher, Michael Jahrer, and Robert Legenstein. Improved neighborhood-based algorithms for large-scale recommender systems. In *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, NETFLIX '08, pages 4:1–4:6, New York, NY, USA, 2008. ACM.

[53] Robert Wetzker, Winfried Umbrath, and Alan Said. A hybrid approach to item recommendation in folksonomies. In *Proceedings of ESAIR '09*, pages 25–29, New York, NY, USA, 2009. ACM.

[54] Bin Xu, Jiajun Bu, Chun Chen, and Deng Cai. An exploration of improving collaborative recommender systems via user-item subgroups. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 21–30, New York, NY, USA, 2012. ACM.

[55] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 267–273, New York, NY, USA, 2003. ACM.

[56] Gui-Rong Xue, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 114–121, New York, NY, USA, 2005. ACM.

[57] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 315–324, New York, NY, USA, 2011. ACM.

[58] Yin Zhu, Yuqiang Chen, Zhongqi Lu, Sinno Jialin Pan, Gui rong Xue, Yong Yu, and Qiang Yang. Heterogeneous transfer learning for image classification. In *In Special Track on AI and the Web, associated with The Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

Appendix A

Basic Concepts

A.1 Clustering

Clustering is the task of grouping a set of objects in a way that objects in the same group, called a cluster, are more similar to each other than to those in other clusters. Several clustering methods has been proposed such as Expectation Maximization(EM), hierarchical clustering, density-based clustering, etc. **K-Means** is one of the most popular clustering method in the literature. K-Means clusters objects into k clusters in which each object belongs to the cluster with the nearest mean. Suppose we have a given data set X consisting of N D-dimensional observations. K-Means employs a set of D-dimensional vectors μ_k, where $k = 1, 2, ..., K$. μ_k represents the mean vector of cluster k. The goal of K-Means is then to find an assignment of data points to clusters, as well as a set of vectors μ_k, such that the sum of the squares of the distances of each data point to its closest mean vector is a minimum [4].

Let's assume data set $X = x_1, x_2, ..., x_N$ with N objects, and binary indicator variables $I_{nk} \in 0, 1$ where:

$$I_{nk} = \begin{cases} 1 & \text{if } x_n \text{ is assigned to cluster } k \\ 0 & otherwise \end{cases} . \tag{A.1}$$

We can then define an objective function that represents the sum of the squares of the distances of each data point to its assigned cluster's mean vector, μ_k, as follows:

$$J = \sum_{n\in\{1,..,N\}} \sum_{k\in\{1,..,K\}} I_{nk}.\|x_n - \mu_k\|^2 \tag{A.2}$$

The optimization goal is to find values for the $\{I_{nk}\}$ and the $\{\mu_k\}$ those which minimize objective function J. This can be done through an iterative procedure in which each iteration involves two successive steps corresponding to successive optimizations with respect to the I_{nk} and the μ_k. First we choose some random initial values for $\{\mu_k\}$. Then in the first phase we minimize J with respect to the $\{I_{nk}\}$ by keeping the $\{\mu_k\}$ fixed. In the second phase we minimize J with respect to the $\{\mu_k\}$, keeping $\{I_{nk}\}$ fixed. This two-stage optimization is then repeated for several iteration until convergence [4].

Curriculum Vitae

Name:	(Seyed) Nima Mirbakhsh
Post-Secondary Education and Degrees:	Semnan University Semnan, Iran 2003 - 2007 BSc Institute for Advanced Studies in Basic Sciences (IASBS) Zanjan, Iran 2008 - 2011 MSc University of Western Ontario London, ON 2011 - 2015 PhD
Honours and Awards:	NSERC Engage Grant, 2014 to 2015.
Related Work Experience:	Teaching Assistant The University of Western Ontario 2011 - 2015 Research Assistant The University of Western Ontario 2011 - 2015 Research Internship Arcane Inc. 2014 - 2015

Publications:

- Nima Mirbakhsh and Charles X. Ling. Improving top-n recommendation for cold-start users via cross-domain information (accepted to publish). the Transactions on Knowledge Discovery from Data (TKDD) , 2015

- Nima Mirbakhsh and Charles X. Ling. Leveraging clustering to improve collaborative filtering. Submitted and under review, 2014.

- Nima Mirbakhsh and Charles X. Ling. Clustering-based factorized collaborative filtering. In Proceedings of the 7th ACM conference on Recommender systems, RecSys 13, pages 315318, New York, NY, USA, 2013. ACM.

- Mohsen Afsharchi, Behrouz H. Far, Arman Didandeh, Nima Mirbakhsh, Common Understanding in a Multi-Agent System Using Ontology-Guided Learning - Knowledge and Information Systems, July 2013, Volume 36, Issue 1, pp 83-120.

- Arman Didandeh, Nima Mirbakhsh, Mohsen Afsharchi, Collaborative Concept Learning Game in a Multi-Agent System, Information System Frontiers, Volume 15, Issue 4, pp 653-676.

- Arman Didandeh, Nima Mirbakhsh, Ali Amiri, Mahmoud Fathi, AVLR-EBP: a Variable Step Size Approach to Speed-up the Convergence of Error Back-Propagation Algorithm - Neural Processing Letters Volume 33, Number 2, 201-214, 2011.

- Nima Mirbakhsh, Arman Didandeh, and Mohsen Afsharchi, Concept Learning Games: The Game of Query and Response - In Proceedings of the IAT 2010, Toronto, Canada, 234-238.

- Nima Mirbakhsh, Arman Didandeh, and Mohsen Afsharchi, Incremental Non-Unanimous Concept Formation through Queried Object Classification - In Proceedings of the 2009

IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2009), Milan, Italy.