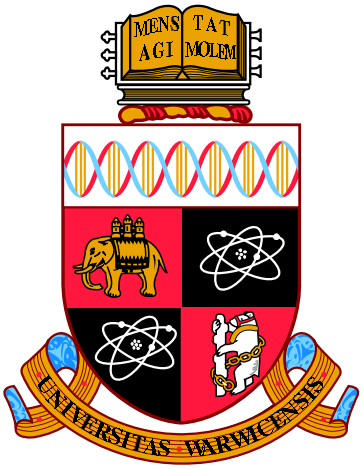


On Reducing the Data Sparsity in Collaborative Filtering Recommender Systems

by

Xin Guan

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Department of Computer Science

April 2017

THE UNIVERSITY OF WARWICK

Contents

Abstract

A recommender system is one of the most common software tools and techniques for generating personalized recommendations. Collaborative filtering, as an effective recommender system approach, predicts a user's preferences (ratings) on an item based on the previous preferences of other users. However, collaborative filtering suffers from the data sparsity problem, that is, the users' preference data on items are usually too few to understand the users true preferences, which makes the recommendation task difficult.

This thesis focuses on approaches to reducing the data sparsity in collaborative filtering recommender systems. Active learning algorithms are effective in reducing the sparsity problem for recommender systems by requesting users to give ratings to some items when they come in. However, this process focuses on new users and is often based on the assumption that a user can provide ratings for any queried items, which is unrealistic and costly. Take movie recommendation for example, to rate a movie that is generated by an active learning strategy, a user has to watch it. On the other hand, the user maybe be frustrated when asked to rate a movie that he/she has not watched. This could lower the customer's confidence and expectation of the recommender system. Instead, an *ESVD* algorithm is proposed which combines classic matrix factorization algorithms with ratings completion inspired by active learning, allowing the system to 'add' ratings automatically through learning. This general framework can be incorporated with different *SVD-based* algorithms such as *SVD++* by proposing the *ESVD++* method. The proposed *EVSD* model is further explored by presenting the *MESVD* approach, which learns the model iteratively, to get more precise prediction results. Two variants of *ESVD* model: *IESVD* and *UESVD* are also proposed to handle the imbalanced datasets that contains more users than items or more items than users, respectively. These algorithms can be seen as pure collaborative filtering algorithms since they do not require human efforts to give ratings. Experimental results show the reduction of the prediction error when compared with collaborative filtering algorithms (matrix factorization).

Secondly, traditional active learning methods only evaluate each user or items independently and only consider the benefits of the elicitations to new users or items, but pay less attention to the effects of the system. in this thesis, the traditional methods are extended by proposing a novel generalized system-driven active learning

framework. Specifically, it focuses on the elicitations of the past users instead of the new users and considers a more general scenario where users repeatedly come back to the system instead of during the sign-up process. In the proposed framework the ratings are elicited by combining the user-focused active learning with item-focused active learning, for the purpose of improving the performance of the whole system. A variety of active learning strategies are evaluated on the proposed framework. Experimental results demonstrate its effectiveness on reducing the sparsity, and then enables improvements on the system performance.

Thirdly, traditional recommender systems suggest items belonging to a single domain, therefore existing research on active learning only applies and evaluates elicitation strategies on a single-domain scenario. Cross-domain recommendation utilizes the knowledge derived from the auxiliary domain(s) with sufficient ratings to alleviate the data sparsity in the target domain. A special case of cross-domain recommendation is multi-domain recommendation that utilizes the shared knowledge across multiple domains to alleviate the data sparsity in all domains. A multi-domain active learning framework is proposed by combining active learning with the cross-domain collaborative filtering algorithm ($RMGM$) in the multi-domain scenarios, in which the sparsity problem can be further alleviated by sharing knowledge among multiple sources, along with the data acquired from users. The proposed algorithms are evaluated on real-world recommender system datasets and experimental results confirmed their effectiveness.

Acknowledgments

First and foremost, I would like to take this opportunity to express my deepest gratitude and respect to my supervisor Prof. Chang-Tsun Li, who constantly gave me support during my Phd time at the University of Warwick. His great personality, unlimited patience and tolerance have educated me a lot more than scientific research.

My parents, Dr. Huaimin Guan, Mrs. Cuihua Wang and my brother, Dr. Yu Guan also deserve my cordial gratitude. Their love, support and encouragement have always been the source of my strength and the reason I have progressed this far.

I wish to express my sincere thankfulness to my annual progress panel members Dr. Victor Sanchez, Dr. Abhir Bhalerao and Dr. Nathan Griffiths for their guidance and valuable suggestions on my PhD progress.

I would also like to thank the colleagues at the department, Dr. Alaa Khadidos, Dr. Ruizhe Li, Dr. Xin Lu, Dr. Xingjie Wei, Dr. Yi Yao, Dr. Xufeng Lin, Mr. Ning Jia, Mr. Roberto Leyva, Mr. Qiang Zhang, Mr. Bo Wang, Mr. Shan Lin, Mr. Ching-Chun Chang, Mr. Yi Hao and Mr. Yijun Quan for their kindness and support.

Last but not least, many thanks to my friends, Dr. Tinghua Duan, Dr. Jiang Wang, Mr. Xiaopeng Cai, Mr. Cheng Li, Mr. Jianxiong Tie, Mr. Lidou Hao, Mr. Guang Chen, Mr. Boyang Peng and Mr. Chengyu Yu, for sharing in my happiest moments, and for genuinely feeling the same.

Declarations

I hereby declare that the work presented in this thesis entitled *On Reducing the Data Sparsity in Collaborative Filtering Recommender Systems* is an original work and has not been submitted to any college, university or any other academic institution for the purpose of obtaining an academic degree.

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Recommender Systems Techniques

Recommender systems have become increasingly common recently and are used by many internet providers. Examples include movie recommendation by *Netflix* [1], web page ranking by Google [2], related product recommendation by Amazon [3], social recommendation by Facebook [4], etc. They provide users with personalized suggestions by predicting the rating or preference that the users would give to an item, and typically apply techniques and methodologies from other neighboring areas such as *Human Computer Interaction* or *Information Retrieval*. In addition, *Data Mining* plays a vital role in recommender systems since the core algorithms in most of these systems can be understood as a particular case of a Data Mining technique [5].

As one of the most common software tools and techniques, recommender systems are used for generating recommendations to users, usually in one of the following ways:

- Collaborative filtering [6] [7] [8] predicts other items the current users might like based on the past knowledge about preferences (usually expressed in ratings) of users for some items. The basic assumption of collaborative filtering

Table 1.1: An example of a rating matrix

User \ Movie	The Godfather	Star Wars	Jurassic Park	Lion King
Joseph	null	null	3	null
Ian	1	5	4	1
Kyle	5	2	3	null
Leonard	4	null	5	3
Jay	null	4	null	2

is that people who agreed in the past will also agree in the future [9]. Pure collaborative filtering approaches take a user-item rating matrix (Table 1.1) as the only input, for predicting how users in the system like a certain items or generating a list of top-N recommendations for users to choose.

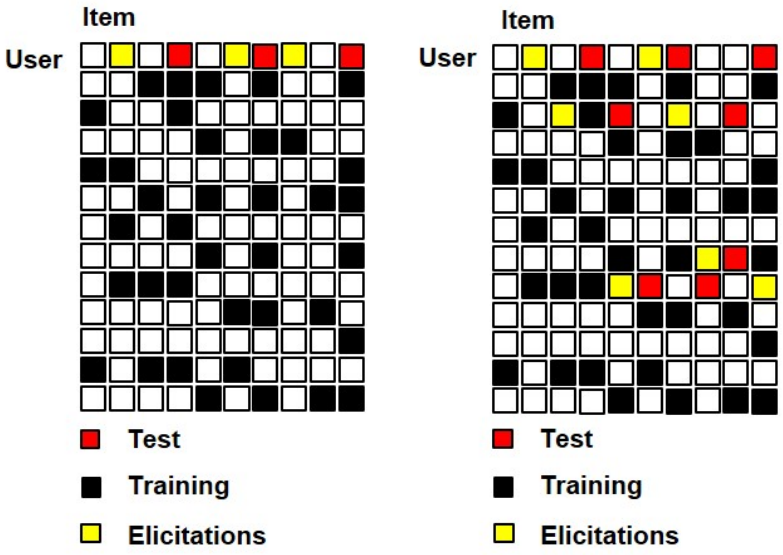
- Content-based algorithms [10] produce recommendations based on items descriptions which can be automatically extracted or manually created, or (and) user profiles that represent the users' interests on items. This type of approaches must rely on the information about items and user preferences, such as genre of a movie, author of a book. However, a large collective of rating history is not required compared with collaborative filtering. Content-based filtering is often used by incorporating with other techniques such as collaborative filtering when additional information is supplied, such as music recommendation [11].

- Knowledge-based algorithms [12] generate recommendations by exploiting explicit user requirements and detailed domain knowledge about item features, reasoning about what items meet the users needs. There are two types of knowledge-based algorithms: Case-based algorithms [13] determine recommendations based on the similarity metrics, trying to find out those descriptions best match the users query, such as the service of personal shopper [14]; constraint-based algorithms [15] make decision on recommendations by exploiting predefined recommender knowledge bases that contain explicit constraints about how to relate a user's requirements with item properties, which is commonly used in financial services [16] and e-tourism [17]. In contrast to

content-based recommender systems, knowledge-based systems rely mainly on externally provided information about the available items.

- Hybrid approaches [18] generate recommendations by combining several algorithms or recommendation components, which are based on the above three base approaches: collaborative filtering and content-based and knowledge-based algorithms. The three main recommendation approaches exploit different sources of information: collaborative filtering algorithms are based on user preferences (i.e. ratings); content-based approaches rely on item features and textual descriptions; knowledge-based exploit external knowledge as the logical rules that map the users requirements onto item features. When multiple sources of information are supplied, building hybrid systems that combine the strengths of different algorithms leads to the improvement of the overall accuracy.

This thesis focuses on collaborative filtering recommender systems because the collaborative filtering algorithm is considered the most important technique, and is widely used in industry, especially in online retail sites to customize the needs for customer, in order to promote additional items and increase sales [1]. In particular, the user-item rating matrix (matrices) is considered to be the only source of information in this thesis. The task is to predict the users' preferences on a specific item, which can be defined as: given a rating matrix $\mathrm{R} \in \mathbb{R}^{m \times n}$ that consists of m users and n items where each rating r_{ij} represents the preference of user i to item j, fill the missing values in R so that the recommender system can recommend the items with the highest predicted ratings in the row $\mathrm{R}_{i,:}$ to the user i.



(a) Active learning for new user (new user problem)

(b) Active learning for all the users

Figure 1.1: An example of active learning

1.2 Active Learning in Recommender Systems

Most collaborative filtering algorithms suffer from the new user problem. That is, when a new user comes in, there is not enough knowledge about this user. As a result, the system will fail to generate proper recommendations given the circumstance. Active learning [19] for recommender systems has been initially proposed for tackling the new user problem. In real-life scenarios, most recommender systems would only ask the user to rate a limited number of items (elicitations) during the sign-up process [20], for better predicting the preferences of the target user (as shown in Figure 1.1(a)).

Apart from the sign-up process, users can give elicitations whenever he or she is motivated, based on the assumption that users would come back to the system regularly [21]. Under this setting, ratings could be elicited (elicitations) from both new users (without training data) and existing users (with training data) in the system by querying them to rate a number of items (as shown in Figure 1.1(b))).

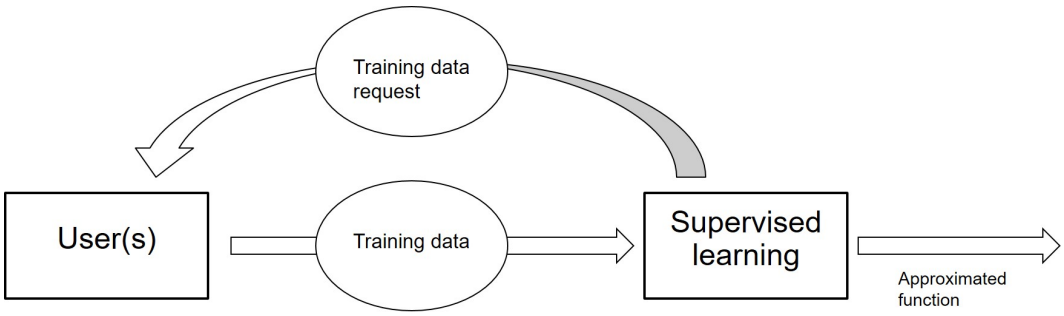


Figure 1.2: Active learning procedure

In both scenarios, the knowledge of the user (or the system) are extended by requesting users for more data (a.k.a. ratings), and then affects the recommendation accuracy for the target users (as shown in Figure 1.2). However, the usefulness of each rating may vary significantly, special techniques (a.k.a. active learning strategies) can be used to intelligently obtain data that better reflects user's preferences and enables to produce better recommendations.

In the review work of [22], a variety of active learning strategies have been analyzed and classified with respect to two distinct dimensions: personalization and hybridization.

- Personalization: personalized strategies query different user for different items based on the characteristics each user has, while non-personalized strategies request all the users to rate the same items.

- Hybridization: single-heuristic strategies are based on one heuristic by utilizing the unique selection rule for both items and users, while combined-heuristic strategies implement multiple selection rules for items and users by aggregating and combining a number of single-heuristic strategies, in order to achieve a range of objectives.

It should be noted that, with more and more ratings being elicited by active learning, not only the cold start problem is addressed, the sparsity problem is also alleviated. As a result, the rating elicitations improve the prediction accuracy of

Table 1.2: An example of cross-domain recommender system

Domains		Joseph	Ian	Kyle	Leonard	Jay	Xin
Book	Harry Potter	3			5		4
	The Hobbit	1			3		5
Movie	The Godfather			2	3	5	2
	Star Wars			5	1	2	5
Music	Let It Be		2			4	2
	Hey Jude		1			3	3
Domains		no overlap			overlap between domains		

the queried users, along with the performance of the whole system.

1.3 Cross-domain Recommender Systems

Traditional recommender systems suggest items belonging to a single domain. Examples include movies in *Netflix*, books in *Book-Crossing*, songs in *Last.fm*, etc. Nowadays, users provide feedback for items of different types (e.g. books, DVDs, etc.) and express their opinions on different social media and different providers (e.g. Amazon, Netflix, etc.). Instead of treating each domain independently, knowledge could be transfered from the source domain to the target domain for better prediction accuracy based on the assumption that information overlap between users and/or items across different domains (Table 1.2), which is referred as cross-domain recommendations.

A domain is a particular field of thought, activity or interest. In the literature [23] researchers have considered distinct notions of domain at four levels:

- Attribute level: same types of items with different values of certain attribute (e.g. comedy and thriller in movie genres).

- Type level: similar types of items, sharing some attributes (e.g. movie and TV shows in Amazon).

- Item level: different types of items (e.g. books in Book-Crossing and movies in Movielens)

- System level: same type of items on different systems (e.g. movies in *Netflix* and *Movielens*)

The goal of cross-domain recommendation is to utilize the knowledge derived from the auxiliary domain(s) with sufficient ratings to alleviate the data sparsity in the target domain. A special case of cross-domain recommendation is multi-domain recommendation that utilize the shared knowledge across multiple domains to alleviate the data sparsity in all domains, when all domains suffer from the data sparsity problem [24].

In this work, the multi-domain recommendation, which utilizes the shared knowledge across multiple domains, is considered at the item level (books in Book-Crossing and movies in *Movielens*&*Netflix*).

1.4 Datasets

In collaborative filtering algorithms, recommendations are generated by exploiting ratings as the source of information. Ratings are collected by asking user's opinion about items on a rating scale, usually in a variety of forms [8]:

- Numerical ratings: such as the 1-5 stars provided in the movie or book recommender systems.

- Binary ratings: that model choices the user to an item, in which the user is simply asked to decide if a specific item is good or bad.

- Ordinal ratings: expressed as the levels of perferences such as [strongly agree, agree, neutral, disagree, strongly disagree], in which the user is asked to select the term that indicates his or her opinions to an item.

- Unary ratings: usually expressed in explicit way, such as purchase, assess, save, delete, etc.

Datasets are formed as a collection of ratings, then are used as benchmarks to evaluate new recommendation algorithms and to compare with other existing algorithms. Datasets are important for training and testing recommender systems, therefore some commonly used datasets are introduced including *MovieLens*, *Netflix*, *EachMovie*, *Book-Crossing*, *Jester* and *Yahoo! Music*.

- *MovieLens* Dataset [25]: a classic recommender system that recommends films to users through collaborative filtering algorithms. There are three datasets of different sizes that are collected by Grouplens Research. The 100K and 1M datasets contain demographic information about the users (age, gender, occupation, zip), while in 10M dataset only user ID is given. The 100K dataset collects 100,000 ratings from 943 users on 1,682 movies. The 1M dataset contains 1,000,209 entered by 6,040 users for 3,900 different movies. The 10M dataset consists of 10,000,054 ratings with 95,580 tags to 10,682 movies provided by 71,567 users. For all three datasets, each user has rated at least 20 movies, and each rating is an integer ranging from 1 to 5 which represents the interests the user has to this movie.

- *Netflix* Dataset [1]: the world's largest online DVD rental service company, that released their dataset collected between October 1998 and December 2005. It consists of over 100 million 5-star ratings of 480,189 users and 17,770 movies.

- *EachMovie* Dataset [26]: a movie recommender system that contains 2.8 million numeric ratings entered by 72,916 users for 1,628 films and video. Each rating ranges from 1 to 6, representing the preferences of the user to the movie.

- *Book-Crossing* Dataset [27]: a book rating dataset containing 1.1 million explicit (expressed on a scale from 1 to 10) and implicit (expressed by 0) ratings from 278,858 users on 271,379 books. In particular, demographic data (location, age) is provided for the users if available, and some content-based information (title, author, year of publication, publisher) is given for the items.

Table 1.3: Comparison of different datasets of recommender systems

Dataset	Domain	Size			Scale
		Users	Items	Ratings	
Movielens 100K	Movie	943	1,682	100,000	1 to 5
Movielens 1M	Movie	6,040	3,900	1,000,209	1 to 5
Movielens 10M	Movie	71,567	10,682	10,000,054	1 to 5
EachMovie	Movie	72,916	1,628	2,811,983	1 to 6
Book-Crossing	Book	278,858	271,379	1, 149, 780	1 to 10
Jester v1	Joke	73, 421	100	4, 000, 000	-10.00 to 10.00
Jester v2	Joke	59, 132	150	1, 700, 000	-10.00 to 10.00
Yahoo! Music	Music	1,000,990	624,961	262,810,175	1 to 5

- *Jester* Dataset [28]: a web-based joke recommender system, developed at University of California, Berkeley. The first *Jester* dataset contains over 4.1 million continuous ratings (-10.00 to +10.00) of 100 jokes from 73,421 users: collected between April 1999 - May 2003, and the second dataset collects over 1.7 million continuous ratings (-10.00 to +10.00) of 150 jokes from 59,132 users: collected between November 2006 - May 2009.

- *Yahoo! Music* Dataset [29]: a personalized internet music recommender system. The dataset consists 262,810,175 ratings of 624,961 music songs by 1,000,990 users collected during 1999-2010. The ratings include one-minute resolution timestamps, allowing refined temporal analysis. Each item and each user have at least 20 ratings in the whole dataset.

Overall, the datasets in real-world recommender systems often consist of large number of ratings that represents users' preferences on items in the form of different ratings scales. This thesis mainly utilizes *Movielens* and *Netflix* datasets for training and test the proposed collaborative filtering algorithms and active learning algorithms in Chapter 3 and Chapter 4. For exploring cross-domain collaborative filtering techniques in Chapter 5, the *Movielens*, *Netflix* and *Book-Crossing* datasets are employed as the input matrices for multi-domain recommendations.

1.5 Evaluation

The quality of a recommender system can be decided based on the results of evaluation. Metric selection depends on the type of collaborative filtering applications. A majority of the work has focused on the evaluation of a recommender systems accuracy, which is also the main task of collaborative filtering recommender systems. To compare the accuracy of different collaborative filtering algorithms, the metrics must be predefined. An accuracy metric empirically measures the difference between a recommender system's predicted ratings and the user's true ratings for a specific item, or between the predicted ranking of items for a user and the user's true ranking of perference: the less, the better. Accordingly, recommendation accuracy metrics are typically classified into three classes: predictive accuracy metrics, classification accuracy metrics and rank accuracy metrics.

- Predictive accuracy metrics are used for measuring how close a recommender system's predictions are to the users true ratings for each movie. Commonly used metrics include Mean Absolute Error (MAE) [30] and its variations.

- Classification accuracy metrics measure the frequency with which a recommender system recommends relevant or irrelevant items for a given user, examples include *Precision* [31], *Recall* [31], *Mean Average Precision (MAP)* [6] and *Receiver Operating Characteristic (ROC)* [6], which of metrics are often used for the top-N recommendation, i.e. the recommender system produce a number of recommendations once a time for the user.

- Rank accuracy metrics extend classification accuracy metrics by taking items relative position in recommendation lists into account. It measures the ability of a collaborative filtering algorithm to produce a ranked recommendation lists for a user that matches the user's ordering of the same items, usually by *half-life utility* [30] or *Normalized Discounted Commulative Gain (NDCG)* [32].

Sometimes even if a recommender system is able to correctly rank users' preferences on items, the system could fail to predict the ratings. On the other hand, the predicted ratings can be used for creating an ordering across the items, for measuring the ability of a recommender system to rank items with respect to user preference. Therefore this thesis is mainly focusing on the collaborative filtering recommendations in terms of prediction task, the commonly used predictive accuracy metrics are introduced. For other collaborative filtering evaluation metrics, see the work of Herlocker et al. in [6] as well as the work of Shani et al. in [33].

Mean Absolute Error (MAE) and *Root Mean Square Error (RMSE)* are two most commonly used predictive accuracy metrics. Given a recommender system, in where r_{ij} is the rating that the ith user gives to the jth item, $\tilde{r}_{ij}$ is the predicted ratings accordingly, and T is the total number of test samples. *MAE* [30] is defined as:

$$MAE = \frac{\sum_{(i,j \in TestSet)} |r_{ij} - \tilde{r}_{ij}|}{T} \tag{1.1}$$

MAE measures the average absolute deviation between a predicted rating and the users true rating. *RMSE* [1] is defined as:

$$\text{RMSE} = \sqrt{\frac{\sum_{(i,j \in TestSet)} (r_{ij} - \tilde{r}_{ij})^2}{T}} \tag{1.2}$$

RMSE amplifies the contributions of the absolute errors between the predictions and the true values, therefore the result has more emphasis on large errors when compared with *MAE*.

This thesis mainly adopts *RMSE* as the evaluation metric for measuring collaborative filtering algorithms.

Table 1.4: An example of the cold start problem

User \ Movie	The Godfather	Star Wars	Jurassic Park	Fast & Furious 8	Lion King
Joseph	null	null	3	null	null
Xin	null	null	null	null	null
Ian	1	5	4	null	1
Kyle	5	2	3	null	null
Leonard	4	null	5	null	3
Jay	null	4	null	null	2

1.6 Chanllenges

1.6.1 Cold Start

Most collaborative filtering algorithms suffer from the cold start problem, which occurs when a new user or item has just entered the system. Since there is no information for the target user or item, the collaborative filtering will fail to generate recommendations. Cold start problem is also known as new user problem (e.g. user Xin in Table 1.4) or new item problem (e.g. movie Fast & Furious 8 in Table 1.4). Most of the research utilizes the hybrid recommendation approach [34], which combines content-based and collaborative filtering, to tackle the cold start problem. However, building model based on hybrid approach is usually complicated, while the improvements is limited. One of the effective solutions is to apply active learning techniques that query the users to rate some specific items during the sign-up process [20].

1.6.2 Sparsity

The sparsity of a rating matrix is defined as [35]:

$$Sparsity = 1 - \frac{\#ratings}{\#users \times \#items} \tag{1.3}$$

where # denotes the total number.

The sparsity problem is the major bottleneck for collaborative filtering algorithms. In most recommender systems, the number of ratings obtained from each

Table 1.5: Sparsity of different datasets of recommender systems

Dataset	Sparsity
Movielens 100K	93.70%
Movielens 1M	95.75%
Movielens 10M	98.69%
EachMovie	97.63%
Book-Crossing	99.998%
Jester v1	45.52%
Jester v2	80.83%
Yahoo! Music	99.96%

user is usually very small compared to the number of available items in the dataset since users are typically reluctant to rate a large amount of items. Therefore the user-item rating matrix used for collaborative filtering will be extremely sparse (as shown in Table 1.5). While most research in the field of recommender systems focus on improving prediction algorithms, even the best algorithm will fail without sufficient data. Take movie recommendation as an example, the movies that have been rated with only few ratings would be recommended rarely, even with high ratings. Also, users with special tastes for movies usually suffer from poor recommendations since similar users are rare in the system.

In this thesis, the cold start problem is not the priority to be concerned since it only arises when a new user or item is added into the systems. Instead, the techniques that tackle the sparsity problem are focused because it happens universally in collaborative filtering recommender systems.

1.7 Research Questions

This thesis addresses the sparsity problem of collaborative filtering algorithms in three aspects, which leads to three research objectives, respectively.

- Recommender systems often apply active learning to handle the cold start and sparsity problem, which sometimes is unrealistic and costly. In the field of col-

laborative filtering recommender systems, I propose to extend the traditional collaborative filtering algorithm (i.e. matrix factorization) with ratings completion. In the proposed method, systems automatically 'add' ratings based on a variety of rules in the framework of matrix factorization algorithm. With the extra generated ratings, the sparsity problem is alleviated and the performance of the recommender system is improved.

- Active learning algorithms enrich the dataset by querying users to label items, often focusing on single user or users, without considering the benefits of the whole system. In the field of active learning collaborative filtering, I propose a general system-driven framework for applying active learning in recommender systems. In the proposed framework, the system queries specific users to rate specific items based on combined rating elicitation strategies. Results suggest its effectiveness in handling the sparsity problem.

- Cross-domain collaborative filtering techniques alleviate the sparsity problem by exploiting knowledge from auxiliary (source) domains. A novel multi-domain active learning framework is proposed by incorporating active learning techniques with cross-domain collaborative filtering algorithms in the multi-domain scenarios. Therefore in each single-domain the sparsity problem can be alleviated by querying users for ratings, aggregating them will further handle the sparsity problem, resulting in further improvements of the prediction accuracy.

1.8 Outline

The rest of this thesis is organized as follow:

Chapter 2 reviews the traditional collaborative filtering algorithms, such as memory-based and model-based collaborative filtering algorithms. Then active learning techniques in collaborative filtering recommender systems are discussed and

summarized based on the characteristics of the elicitation strategies. The state-of-the-art cross-domain collaborative filtering algorithms are presented as well.

Chapter 3 first briefly introduces the collaborative filtering task. Then the matrix factorization method [36] is presented as one of the most commonly used collaborative filtering algorithms in recent years. Based on this framework, the proposed *Enhance SVD* method and its variations are introduced by exploiting ratings completion, along with the corresponding experimental analysis. A summary of the proposed algorithms is provided at last.

Chapter 4 introduces the active learning techniques used in collaborative filtering recommender systems and their limitations. The proposed active learning framework is shown by adding constraints to the users. Performance analysis and comparison with traditional active learning are demonstrated at last.

Chapter 5 first introduces the cross-domain techniques, especially the *R-MGM* model that used in multi-domain scenario. Then active learning for multi-domain recommendations is introduced by incorporating active learning techniques with *RMGM* model. Comprehensive evaluations demonstrate the advantages of the proposed framework.

Chapter 6 summarizes the achievements of this thesis and presents some future work.

Chapter 2

Literature Review

2.1 Collaborative Filtering Algorithms for Recommender Systems

Collaborative filtering is a method that makes recommendations by using ratings given to items by users as the only source of information. It was first proposed by Goldberg el at. [9]. They built a collaborative filtering system that allowed users to annotate messages for filtering emails. This work was proved to be effective by involving human activity in the filtering process in contrast to content based filtering. Later collaborative filtering algorithms have been used widely because of its applicability in many domains. Examples include:

- Hill et al. [37] compared the user's ratings of videos with others to find people with similar interests and gave recommendations based on the ratings that similar people have rated in the video recommendation.

- Shardanand et al. [38] used collaborative filtering for providing suggestions to the user based on similarities between the interest profile of that user and those of other users. Based on this technique they designed a system called *Ringo* which makes personalized recommendations for music albums and artists.

- Grouplens group [39] designed a collaborative filtering system for *Usernet* news to allow user to rate articles. They demonstrated that collaborative filtering could be implemented for predicting ratings to each user.

In recent years collaborative filtering has become the most prominent approach to generating recommendations. Various algorithms have been proposed and evaluated on real-world and artificial test data. Empirical studies such as [34] and [30] categorized the collaborative filtering algorithms into two classes: memory-based algorithms and model-based algorithms, as detailed in the following two subsections.

2.1.1 Memory-Based Collaborative Filtering

Memory-based algorithms [30] [40] predict ratings for the user based on the entire collection of previously rated items by the users. Therefore prediction is computed as an aggregation of the ratings regarding other users that are usually chosen based on the similarity for the same items. Early research on the algorithms of recommender systems were focused on the neighbourhood models [38] [39] [41]. Neighbourhood models give predictions based on the similarity relationships among either users or items. Generally, they select a number of similar users or items based on a certain similarity measure. Then the prediction is computed based on the ratings of their neighbours. Memory-based algorithms can be classified as user-based or item-based depending on whether the process of searching for neighbours focuses on users or items.

2.1.1.1 User-Based Algorithms

Because some users are likely to prefer the same items with the same taste, user-based methods only consider those users similar to the target user instead of using information from all users. User-based algorithm were first proposed by Resnick et al. in [41], and they used *Pearson* correlation coefficient as the similarity measure for selecting the neighbours of the target user. In their work, *Max Number*

of Neighbours strategy was proposed for neighbour selection, i.e. a number of the most similar users are selected as the neighbours of the target user. A large number of the neighbours tend to introduce too much noise because the users with small correlations are also chosen as neighbours, while a small number of neighbours is also likely to result in poor prediction accuracy because highly correlated users are excluded from the neighbours. The prediction was calculated by the aggregation of each neighbour which is weighted by his/her similarity with the active user. Subsequently, another neighbourhood selection strategy was proposed by Shardanandi et al. [38]. They named it *Correlation Threshold* strategy, which only selects the users by thresholding the similarities. It limits the neighbour to contain good correlations. However, only a small number of the neighbours will be chosen for some target users who have less high correlated neighbours.

Most research on user-based collaborative filtering algorithms are focused on various approaches to computing the similarity measure between users. Shardanand et al. [42] proposed to use *Mean Squared Difference* as the similarity measure between users based on the mean difference of the items that both users have rated. Later the same authors [38] took into consideration that the ratings consisting of both positive numbers and negative numbers, and proposed the *Constrained Pearson* that uses the median of ratings instead of the mean rating in the *Pearson* correlation coefficient as the similarity measure. Another very common measure is cosine similarity which measures the cosine between the vectors of two users [30]. Cosine similarity is normally used in information retrieval and text mining, and has been shown to produce better results in item-based recommender systems compared with user-based recommender systems. Herlock et al. [43] further improved the accuracy of user-based algorithms by adding fine-grained neighbor weighting factors. They [44] also proposed the *Z-score Normalization* which adds a normalization function before weighting the user's ratings according to similarity because of the differences in rating distributions among users.

Recent research takes side information into account in more complicated scenarios to explore the further potential of the user-based algorithms. Melville et al. [7] applied prediction by content-based algorithms (based on the content of the items and user profiles) to convert a sparse user rating matrix into a full rating matrix, and employed user-based collaborative filtering for recommendation. Results showed that the accuracy of recommendation is improved by extending collaborative filtering with content information of items. To alleviate the sparsity problem of rating matrix where finding similar users is often failed, Massa et al. in [45] and [46] proposed to propagate trust over the trust social network and inferred the trust weight instead of the similarity weight, respectively. In [47], the geo-tags were used for improving the user-based collaborative filtering. Specifically, the similarity between two users were calculated by their geo-tag distributions based on *Gaussian* kernel convolution. The geo-tags of the most similar users were chosen, then combined to re-rank the popular locations in the target city for personalized location prediction. [48] used both social network and the user-contributed tags as the side information to generate the similarities between users for ratings prediction.

2.1.1.2 Item-based Algorithms

In contrast to user-based methods, item-based algorithms use similar items instead of users as neighbors for the aggregation. Most similarity measures used in user-based methods work for item-based algorithms. Cosine similarity [30] or adjusted cosine similarity [49] are commonly used and have been proved to be effective in item-based scenarios.

In previous works, weights were often calculated by arbitrary similarity functions, mainly through trial and error. Bell and Koren [50] pointed out that the traditional neighborhood-based methods do not account for interactions among neighbors. Take a movie recommendation for example, a series of movies (such as Harry Potter series) are highly correlated of each other. An algorithm that ignores the

similarity of these movies when determining their interpolation weights, may end up essentially multiple counting the information provided by the group. They [50] proposed to apply optimization to find the weights by minimizing the squared error between an item's rating and ratings of its neighbours. It was considered to be not only a more principled approach, but by deriving weights simultaneously, interaction effects was overcame. Subsequently, Koren [51] further proposed a more accurate neighbourhood model by considering what the user rated as explicit information, but also what he or she did not rate as implicit information. Because both latent factor model and neighbourhood model have their own merits and drawbacks, by combining them together, they obtained an integrated model by allowing them to enrich each other [51].

Side information was also studied when collaborating with the item-based algorithms. *TrustWalker* [52] has been proposed by integrating a random walk model into item-based collaborative filtering with trust information. In the work of [18], the authors used semantic ratings obtained from the knowledge-based part (knowledge of how these items meet a user's needs) of the system to improve collaborative filtering for the purpose of recommending restaurants. Tso et al. [53] proposed to compute the similarities between items using the combination of attribute information and rating based similarities. In [54], Firan et al. introduced an algorithm that uses tags to recommend users' interested songs and studied the difference between collaborative filtering recommendations based on tag profiles and recommendations based on song/track profiles (content-based). Later Tso et al. [55] proposed a similarity fusion algorithm that calculates the user-user (or item-item) similarity based on both tags and ratings within memory-based collaborative filtering. *Tagommenders* [56] were proposed as a group of tag-based recommendation algorithms that predict users' preferences for items based on their inferred preferences for tags. Liang et al. [57] extracted the semantic meaning of each tag by exploring the multiple relationships among users, items and tags, and a weighting scheme was applied

based on the semantic meaning of each tag in the traditional memory-based framework. In addition to tags, geo-tags have also been exploited. Users' frequented shops were employed as input to the item-based collaborative filtering algorithm for shop recommendation [58]; Horozov et al. [59] made use of location information as a key criterion for restaurant recommendation. Studies showed that incorporating side information would enhance the recommendation quality of the memory-based collaborative filtering algorithms.

2.1.1.3 User-Based vs Item-Based

Both user-based and item-based methods need to identify the nearest neighbours to the target sample based on a certain similarity measure either by user's or item's perspective. The choice of user-based or item-based algorithms depends on the characteristics of the domain. For most recommender systems, there typically are many more users than items, and new users come in much more frequently than new items, so it is easier to compute all pairs of item-item similarities. Another consideration is that item-based algorithms can be helpful in offering an explanation as to why an item was recommended, since similar users have purchased an item is less persuasive than arguing that a given item is recommended because it is similar to other items purchased in the past. However, in some domains such as the context of news, the item dimension changes much faster than the user based, therefore the system should favour the user-based approach.

2.1.2 Model-Based Collaborative Filtering

Another category of collaborative filtering is model-based methods. Model-based algorithms predict ratings based on the models which are learnt from the collection of ratings. Recent studies showed that a lot of model-based algorithms are related to machine learning. Examples include:

- Both Billsus et al. [60] and Sarwar et al. [61] employed *Singular Value Decom-*

position (*SVD*) to predict ratings for recommender systems.

- In [62] and [63], clustering was applied for generating recommendations.

- Billsus et al. [60] predicted items for users based on *Neural Networks*.

- In the work of [30], Breese et al. concluded the empirical collaborative filtering algorithms, and employed *Bayes Networks* to model the conditional probability between items for recommendation.

- In addition, Aggarwal et al. [64] proposed a new graph-based approach to collaborative filtering.

- *Principal Component Analysis* (*PCA*) was also applied to facilitate dimensionality reduction and rapid computation for rating predictions [28].

- In [65], probabilistic factor analysis model for collaborative filtering was presented.

Apart from the traditional machine learning methods, *Latent Semantic Analysis (LSA)* [66] has been widely used in natural language processing for analyzing relationships between documents, which is also deeply exploited in collaborative filtering scenarios. Hofmann et al. [67] introduced a statistical method to collaborative filtering which employ latent class models that based on observed preference behaviors for predicting user preferences. Subsequently, the same authors also proposed a statistical algorithm called *Probabilistic Latent Semantic Analysis* (*PLSA*) [68] for the analysis of two-mode and co-occurrence data (user-item pairs). This technique was later employed and proved to be effective in collaborative filtering scenarios by [69]. Popescul et al. [70] extended *PLSA* to incorporate three-way co-occurrence data among users, items, and item content. They showed that combining collaborative filtering and content-based filtering in this manner generate better quality recommendations. In the work of [71], Wang et al. presented a generalized latent

semantic analysis called *M-LSA*, which conducts *LSA* by incorporating all pairwise co-occurrences among multiple types of objects. *M-LSA* identifies the most salient concepts among the co-occurrence data and represents all the objects in a unified semantic space, showed its effectiveness in utilizing all the information on a multiple-type graph. Later, Wetzker et al. [72] further extended *PLSA* by integrating item-tag relations with the rating matrix for item recommendation. Likewise, Rattenbury et al. [73] exploited place semantics from tags associated with geo-tagged images on social media like *Flickr* (an image hosting and video hosting website) based on the frame work of *PLSA*. In [74], Yin et al. concluded various geographical topic and proposed an algorithm called latent geographical topic analysis by integrating location, text or both for discovering the topics representing a region. Hong et al. [75] presented a new algorithm to discover geographical topics from geo-tagged Twitter messages for location recommendation.

Recently, latent foctor model such as Matrix Factorization (*MF*) techniques have widely spread due to the promising performances they achieves and the good scalability which can be extended to incorporate additional information into recommender systems. Simon Funk [76] proposed *regularized SVD* for collaborative filtering on *Netflix* data. It decomposes the original rating matrix into the products of the side feature matrices, therefore each user's rating is composed of the sum of preferences about the various latent factors of that movie. Since its publication, several improvements of *SVD* algorithms have been proposed in this same context. Peterek [77] improved *regularized SVD* by adding user and item biases, and also post-processed results from *SVD* with *KNN* and kernel ridge regression. Koren [51] proposed the *SVD++* where the implicit feedback (modeled as the items a user has rated) is taken into account. He also merged matrix factorization model with an improved neighborhood model which proved to be effective [51]. Apart from matrix factorization, *Restricted Boltzmann Machines (RBM)* was used for collaborative filtering recommender systems by Salakhutdinov et al. [78]. And it was extended by

incorporating item content features into the model by Gunawardana et al. [79]. Koren also has successfully combined the matrix factorization algorithms with *RBM* in [80]. The early research on time-dependent collaborative filtering was done by Ding et al. [81]. Later Koren [82] integrated the time information into the matrix factorization model for better prediction performance. Subsequently Liu et al. [83] further proposed an incremental version of this work for online recommendation over time.

Another category of research based on matrix factorization is focused on joint factorization with side information instead of factorizing solely U-I matrix. Singh et al. [84] presented *Collective Matrix Factorization (CMF)* which simultaneously factorizes multiple related matrices including the U-I matrix and matrices containing the side information. In early research Salakhutdinov et al. [85] proposed the *Probabilistic Matrix Factorization (PMF)* model which scales linearly with the number of observations, Ma et al. [86] jointly factorized U-I matrix and social trust network based on the same framework. Later Ma et al. [87] implemented joint factorization on U-I matrix, social trust network and social distrust network (by adding penalties to users who are similar to their distrustees). In [88] the same authors proposed a novel probabilistic factor analysis framework, which takes into account of both the users' tastes and their trusted friends' favors together. Later they [89] extended their previous work from only exploiting social trust relationship to exploiting both explicit and implicit social relationships. Social networks and social tags were also exploited by them [90] though a factor analysis approach based on probabilistic matrix factorization. In [91], Ma et al. presented a matrix factorization framework with social regularization which adds social constraints on social-based recommender systems. Apart from social network with matrix factorization listed above, other types of information have also been exploited for improving performance of recommender systems. Zhen et al. [92] proposed joint factorization of the U-I matrix and the tag based user-user similarity matrix by taking tags information into consideration. Shi

et al. [93] jointly factorized the U-I matrix and the mood-specific movie similarity matrix for generating mood-specific movie recommendations. They also jointly factorized the user-landmark matrix and the category-landmark matrix by utilizing geo-tags from photo sharing sites for personalized landmark recommendation [94]. In the work of [95], Zheng et al. proposed an algorithm which jointly factorizes the user's activity correlation matrix, the location correlation matrix, and the location-activity matrix for both location recommendation and activity recommendation. In summary, *CMF*-based models discover the latent representations of different entities by decomposing the relations of each paired entities, which generate more precise recommendations than single factorization models.

Regression-based latent factor models, which was proposed by Agarwal et al. [96], has also been widely used for collaborative filtering. It integrates attributes of both users and items with U-I preference data into a generalized linear model for preference prediction. Later social trust relationship of users were also integrated into the same framework of regression-based latent factor models by Jamali et al. [97]. *fLDA* proposed by Agarwal et al. [98] applied *Latent Dirichlet allocation (LDA)* [99] to regularize the matrix factorization model where side information can be represented in the form of a bag of words (i.e., with statistics of the occurrences of individual words). Agarwal et al. also proposed *Localized Matrix Factorization (LMF)* [100] which makes use of different types of side information by employing local latent factors for each entity. The authors showed that *LMF* overcomes the drawback of *CMF* [84] that uses only global latent factors for each entity, which often results in severe bias due to unbalanced information sources.

Another category of algorithms make use of *Tensor Factorization (TF)* [101] for recommender systems, which has been widely used in the field of signal processing, computer vision, graph analysis, etc. In *Tensor Factorization*, the data are taken in the form of [user, item, interaction context, rating] instead of [user, item, rating] for the common U-I matrix. *Tucker* model and *CANDECOMP/PARAFAC*

(CP) model are two most commonly used Tensor Factorization models [101]. *Tucker* model decomposes a tensor into a core tensor multiplied by a factor matrix with each mode, while *CP* model decomposes a tensor as a sum of rank-one tensors [101]. Tag information has been integrated into the Tucker model with U-I matrix for the purpose of item recommendation by Xu et al. [102], tag recommendation by Symeonidis et al. [103] and both by the same authors [104]. Rendle et al. [105] also proposed an algorithm for tag recommendation based on Tucker model with a pairwise ranking criterion that optimize the latent factor of users, items and tags. Subsequently the *Pairwise Interaction Tensor Factorization (PITF)* model was proposed by modeling the pairwise interactions between users, items and tags in the *Tucker* model framework for tag recommendation. Xiong et al. [106] proposed *Probabilistic Tensor Factorization (PTF)* model which combines *PMF* [85] with *CP* model for integrating time information into U-I matrix for the purpose of item recommendation. Similar to this work, Moghaddam et al. [107] proposed *Extended Tensor Factorization (ETF)* model that combine *PMF* [85] with the *Tucker* model for view recommendation. Overall, tensor factorization methods discover the latent representations of different entities by decomposing the relations of all entities simultaneously, which suits to the case of incorporating interaction-associated information that are directly related to the event of a user interacting with an item [108].

Apart from *Tensor Factorization*, *Factorization Machines (FM)* proposed by Rendle [109] has also drawn a lot of attention. It models all interactions between variables with factorized parameters, therefore combines the advantages of *Support Vector Machines (SVM)* and factorization models. In [110], the same author modeled contextual information and provided context-aware rating predictions based on the same framework of *FM*. In addition, Rendle [111] showed that *FM* can recover many other models just by feature engineering, such as the *Nearest Neighbor Models* [50], the *SVD++* model [51] the *PITF* model [112], regression-based latent factor models [96] etc. Later Nguyen et al. [113] developed a probabilistic algo-

rithm based on *FM* for context-aware recommendation using Gaussian processes. Loni et al. [114] employed *FM* for *Cross-Domain Collaborative Filtering (CDCF)* by allowing interaction information from an auxiliary domain to inform recommendation in a target domain. In contrast to *TF* [101], *FM* allows the modeling of higher-order interactions in a way different from *TF* and thus provides another promising framework for incorporating multiple interaction-associated information to learn recommender system models.

So the commonly used model-based collaborative filtering algorithms are summarized based on how the recommendation model is learned. In next section the advantages and disadvantages of memory-based and model-based algorithms are discussed.

2.1.3 Memory-Based VS Model-Based

Memory-based and model-based algorithms are compared based on three aspects:

1. Explanation: memory-based algorithms such as neighbourhood methods concentrate on the relationship between items or users. So they are good at detecting localized relationships, which can be used as the explainations of the recommendations for users in the systems. While model-based algorithms such as latent factor models try to explain each user's rating by the latent factors of items. Although they give an intuitive rationale for recommendations, the explainations are less compelling.

2. Scalability: memorey-based algorithms have better scalability for handling the cold start problem that the database keeps growing as new users or items continue to be added, but the scalability is limited for large datasets. Most model-based algorithms have to re-train the parameters to the model, and they have trade-off between prediction performance and scalability.

3. Prediction: though memory-based algorithms have good prediction accuracy

for dense datasets, the performance decrease when data are sparse. And it cannot recommend for new users and items. In most recommender systems where ratings matrices are extremely sparse, model-based approaches such as matrix factorization algorithms better address the sparsity and achieve more promising performance.

2.2 Active Learning in Collaborative Filtering Recommender Systems

The quality of the prediction algorithms affect the accuracy and efficiency of collaborative filtering recommender systems given a certain amount of data, hence the collaborative filtering algotihms are summarized in Section 2.1. Apart from the prediction algorithms, the accuracy of collaborative filtering recommender systems also rely on the knowledge that users provided to items (e.g. ratings). Generally, the more informative ratings are obtained, the better performance recommender systems can achieve. However, most recommender systems suffer from the sparsity problem, i.e. the rating matrices are extremely sparse since users are often reluctant to rate a large amount of items. Another challenge of recommender systems is the new-user problem: when a user comes in, it is difficult to give proper suggestions since the system has little knowledge about the target user. Therefore, active learning is widely used for tackling the problem of obtaining high quality data that better represents the preferences of users.

Early research on active learning in collaborative filtering recommender systems focused on reducing the uncertainty of user's opinions. Merialdo et al. [115] first proposed to use *Entropy* and *Variance* as the elicitation strategies based on the framework of neighbourhood algorithm. In their work, the items with the largest entropy or variance were selected for the new users to rate, for the purpose of reducing the uncertainty of user's preferences. Then the ratings of the target users are

calculated based on the neighbourhood method with rating elicitations in the training set. And the performance was evaluated by the improvement of Mean Absolute Error (MAE) against the number of training ratings over *Random* selection strategy. It showed that through this smart selection the recommender system could achieve better performance for a certain amount of ratings required from the user, or reduce the amount of elicitations to reach to the given performance when compared with random selection. Later Boutilier et al. [116] proposed acquiring ratings based on the expected value of information to find the most informative items. Rashid et al. [117] further explored the *Entropy* strategy by proposing the *Entropy0* strategy, where the missing values are considered be 0 as a single category. *Uncertain-Based* strategies such as *Variance*, *Entropy* or *Entropy0* select items with controversial or diverse ratings. However these strategies only reduce the uncertainty of the selected item, Rubens et al. [118] proposed an *Influence-Based* strategy which selects items with the highest influence that reduce the uncertainty over all items (based on the sum of prediction difference between the target item and all the items). Likewise, later in the work of [119], *Impact-Based* strategy was proposed which selects items that have the highest impact on the prediction of other ratings (based on the number of influenced predictions through four-node path in graph-based representation).

Another group of strategies focused on selecting items that are more likely to be familiar to the target user. Such as *Popularity* proposed by Rashid et al. [20], where items with the largest number of ratings are preferred. And *Item-Item Personalized* [20] which presents movies using any strategies until the user has given at least one rating, then selects items that the user is likely to have seen by computing similarity between items. Golbandi et al. [120] introduced the *Coverage* strategy. It selects items with the largest coverage, which is defined as the total number of users who co-rated both the selected item and any other items. In the work of [121], Elahi et al. proposed the *Binary Prediction* strategy that transforms the rating matrix into the $[0, 1]$ binary matrix where known ratings are set to be 1

and unknown ratings are set to be 0. And the items with the highest prediction score are elicited, which are supposed to have the highest probability to be rated by the user. Later in [122], the same authors extended the *Binary Prediction* strategy to the *Personality-Based Binary Prediction* strategy by incorporating side information such as gender, age group and the scores for the *Big Five* personality traits.

Active learning strategies that improve the prediction accuracy of the recommendations are also discussed. Golbandi et al. [120] proposed the *GreedyExtend* strategy, and the items that minimize the *RMSE* of the predictions on the training set are selected. The *Highest and Lowest Predicted* strategies: items with the highest or lowest predicted ratings are chosen. The motivation behind is that the items with the highest or lowest ratings are supposed to be the most liked or disliked movies for this user, which may also influence the user to rate them [123]. In [117], *Information Gain through Clustered Neighbours (IGCN)* was proposed based on decision trees where each node is labelled by a particular item. Users are clustered into groups with similar profiles and items with the largest information gain are elicited in different stages: the first one is non-personalized step, where item with the largest information gain computed by considering all users are elicited by the new user until he or she has rated to a threshold number of items; and the second one is personalized step, where only the best neighbours of the target users are used to compute the information gain as the new criteria for rating elicitation. That is to say, the items that provided the highest information gain for correctly classifying the users in the right cluster are selected. Golbandi et al. [124] extended this work by proposing an adaptive strategy based on decision trees. Specifically, each node is labelled by a particular item and users are divided into three groups based on their possible evaluations on the target item: *Lovers*, *Haters* and *Unknowns*. Then items that minimize the squared error of splitting subsets are selected as rating elicitations. Later Zhou et al. [125] combined the decision-tree based interview model and the matrix factorization model into a single framework for cold start recommendation

which has been proved to be effective. Liu et al. [126] presented the *Representative-Based* strategy that selects a subset of items that represents the whole catalogue based on a certain error criteria.

Some strategies hybridize single strategies in order to achieve a range of objectives. In [20], Rashid et al. proposed the *Popularity*Entropy* strategy, which considers both popularity and entropy; and the *Log(Popularity)*Entropy* strategy, which takes the log of the ratings that linearized popularity, making it a better match for entropy. The same authors further extended their work by proposing the *Harmonic mean of Entropy and Logarithm of rating Frequency (HELF)* strategy [117], which finds items that are familiar by others and with high variability. Likewise, Golbandi et al. [120] introduced the *Squrt(Popularity)*Variance* strategy that finds items with diverse and a large number of ratings. The *Voting* strategy, which considers the overall effect of previous methods, was also proposed by the same authors [120].

Another group of approaches elicited ratings for items based on the prediction model. [127] used Bayesian analysis for active learning in Bayesian Networks. Based on this work, Jin et al. [128] proposed to model active learning for collaborative filtering as Bayesian process by taking into account of the posterior distribution in the framework of aspect model. Previous work are based on the assumption that a user can provide rating for any quired items, Harpale et al. [129] further extended Jin et al.'s work [128] by incorporating an estimate of the probability that a user is able to provide rating based on the same framework of aspect model. Matrix Factorization has drawn a lot of attentions recently. Based on this framework, Karimi et al. [130] introduced the *MinNorm* strategy, which obtains latent vector of each item by *Matrix Factorization* approach and selects items whose corresponding vectors have the minimum *Euclidean* norm. Therefore, the rating elicitations try to avoid large change of the latent factors, which can keep the prediction model stable. The *MinRating* strategy that selects items with lowest predictive score was also

presented. Then Karimi et al. [130] further proposed the *Non-Myopic* strategy that combines the *MinRating* and *MinNorm* strategies based on the same framework.

In previous works, ratings were only elicited during the sign-up process. Carenini et al. [21] pointed it out that users can give elicitations whenever she or he is motivated, therefore they presented the *Conversational and Collaborative Interaction* model where ratings could be elicited from both new users and existing users. The author also proposed the item-focused approach that elicits ratings to improve the rating prediction for a specific item. Elahi et al. [131] suggested that the rating elicitations to users not only improve the prediction of the target user but also help the system to give suggestions for other users. In this work the authors evaluated the active learning strategies in the system wide perspective to test how different elicitation strategies affect the performance of the whole system.

It is shown that different strategies can improve different aspects of the recommendation quality, such as rating prediction accuracy measured by Mean Absolute Error (MAE)/Root Mean Square Error (RMSE), ranking quality measured by Normalized Discounted Cumulative Gain (NDCG)/Mean Average Precision (MAP), number of ratings acquired. In addition to the evaluation measures, the choice of the best strategies also depend on the stages of the rating elicitation process and the dataset.

2.3 Cross-domain Collaborative Filtering for Recommender Systerms

The sparsity problem in recommender systems is a major bottleneck for most collaborative filtering methods. Apart from the active learning algorithms which enrich the dataset by querying users to label items (Section 2.2), many research [132] [133] [134] try to alleviate the sparsity problem by cross-domain recommender systems. Cross-domain recommender system has recured a hot research attention in recent years.

Unlike single-domain which treats each domain independently, cross-domain aims to improve recommendation on a target domain by exploiting knowledge from auxiliary (source) domains that contain abundant user preference data.

There are two primary types of cross-domain approaches, based on how knowledge from the auxiliary domain is exploited: either by aggregating knowledge from both the auxiliary and target domains or transferring knowledge from the auxiliary domain to the target domain.

2.3.1 Aggregating Knowledge

This section reviews the cross domain collaborative filtering approaches which aggregates knowledge from both the auxiliary and target domains, in order to generate recommendations for the target domain. It can obtained by merging user preferences (Section 2.3.1.1), by mediating user modeling data (Section 2.3.1.2) or by combining recommendations (Section 2.3.1.3).

2.3.1.1 Aggregating Knowledge: Merging User Preferences

Merging user preferences from different domain is the most direct way to tackle the cross-domain recommendation problem.

In [135], Winoto et al. pointed out that human preferences may span across multiple domains, therefore the users' consumption behaviors on related items from different domains can be utilized to improve recommendations. They explored the interests of the users in cross-domain scenarios through various statistical analysis and computational analysis based on the traditional collaborative filtering approach. Their extensive analysis shows that the recommendation accuracy is most influenced by the closeness between the crossed domains.

Nakatsuji et al. [136] presented an algorithm that builds *Domain-Specific-User Graphs (DSUGs)* whose nodes (associated with users) are linked by weighted edges that reflect user similarity. *DSUGs* are connected via the users who rated

items in multiple domains or via the users who share social connections, to create a *Cross-Domain-User Graph (CDUG)*. By employing random walk on the *CDUG*, the items that are favoured by the users associated with the extracted nodes are obtained. Through this method the authors try to identify items that the user is interested in but lie in other domains that the user has not accessed before.

In the work of [114], Loni et al. proposed an approach to encode domain-specific knowledge in terms of real-valued feature vectors and allow interaction information from an auxiliary domain to inform recommendation in a target domain. Therefore, Factorization Machines [109] was utilized for incorporating additional knowledge from auxiliary domains to improve prediction accuracy in a target domain in the cross-domain collaborative filtering scenario.

Cross-domain collaborative filtering by merging user preferences is the simplest method, and it works well for new-user problem and facilitates explanation for the recommendations. However, it requires user-overlap between the auxiliary and target domains.

2.3.1.2 Aggregating Knowledge: Mediating User Modeling Data

Mediating user modeling data is another main method for generating recommendations in cross-domain collaborative filtering.

An early approach for cross-domain recommendation through mediation was proposed by Berkovsky et al.. In [137], the authors presented several mediation approaches by aggregating vectors of users' ratings in different collaborative filtering domains: exchange of ratings, exchange of user neighborhoods, exchange of user similarities, and exchange of recommendations. Experimental results showed that the mediation of user modeling data can improve the prediction accuracy.

Later Shapira et al. [138] proposed an approach that uses multi-domain data from social networks (Facebook) to produce the set of candidate nearest neighbours. Several weighting schemes was discussed along with several metrics and recommen-

dation tasks.

In the work of [139], Pan et al. specified uncertain ratings as a range or rating distribution that are estimated by various non-preference data. They proposed an approach called *Transfer by Integrative Factorization (TIF)* that integrate uncertain ratings in the auxiliary domain as additional constraints of the matrix factorization in the target domain. Corresponding experimental results demonstrates its advantages in efficiency and effectiveness of collaborative filtering by incorporating the uncertain ratings from the auxiliary domain.

In summary, mediating user modeling data can achieve good accuracy and maybe suit to the new-user problem. However, either user-overlap or item-overlap between the auxiliary and target domains is needed.

2.3.1.3 Aggregating Knowledge: Combining Recommendations

The idea of combining recommendations was referred to the work of [140], Berkovsky et al. proposed to utilize user modeling data from multiple sources for handling the sparsity problem. Specifically, this paper exploited a content-dependent partitioning method where ratings are partitioned into multiple domains based on the genre of the movie. They showed that the accuracy of the generated predictions is improved by aggregating recommendations of each single domain for the target domain. Givon et al. [141] further explore the recommendations combination by proposing a weighted aggregation method in the book recommendation scenario.

Overall, combining recommendations of different system is easy to implemented, and it also increases the diversity of the training data. But it is difficult to tune weights assigned to recommendations coming from different domains, and the overlap of users is required.

2.3.2 Transferring Knowledge

This section reviews the cross domain collaborative filtering approaches that transfer knowledge between domains. It can be done by linking different domains (Section 2.3.2.1), by sharing latent factors (Section 2.3.2.2) or by transferring rating patterns (Section 2.3.2.3).

2.3.2.1 Transferring Knowledge: Linking Domains

The link between different domains is common, such as comedy movies and humorous books. Therefore many works try to solve the cross-domain recommender system by linking domains.

Zhang et al. [24] considered a multiple domain scenario which learns different collaborative filtering tasks simultaneously (a.k.a. multi-domain collaborative filtering). To solve multi-domain problems, they proposed a probilistic framework which learns each single-domain based on the framework of PMF [85] and allows the knowledge to be transferred adaptively across different domains by learning the correlation between domains.

Cao et al. [142] refer to the recommendation problem as a link prediction task which is defined as a problem that predicts the existence of a link between two entities [143]. Similar to Zhang et al.'s work [24], a more complicated scenario where multiple link prediction tasks from different domains is considered as the *Collective Link Rrediction (CLP)* problem. To solve the *CLP* problem, they proposed a Bayesian framework that learns the correlation between domains adaptively and transfers the shared knowledge among similar tasks, which result in the improvements of the performance for all recommendation tasks.

Shi et al. [144] presented an algorithm called *Tag-induced Cross-Domain Collaborative Filtering (TagCDCF)*, which learns each single-domain with tag-based similarities between user pairs and item pairs as constraints based on framework of matrix factorization method [36], and exploits user-contributed tags in multi-domain

scenarios to learn the links between domains.

In [145], Mirbakhsh et al. proposed an approach that transfers the knowledge in two levels: the traditional user-item level and the new cluster level. Then a cross-domain coarse matrix is defined by capturing the common preferences between clusters of users and cluster of items in same or different domains. Therefore the missing ratings in the cluster-level can be replaced by the observed ratings in the coarse matrix, for the purpose of reducing the sparsity of rating matrices. At last the clustering-based matrix factorization is implemented by aggregating the recommendations from these two levels, which shows promising improvements for all users, especially for cold start users.

In total, transferring knowledge by linking domains does not require user or item overlap between domains, and it can be incorporated with other techniques. But it is difficult to generalize and often designed for particular cross-domain scenarios.

2.3.2.2 Transferring Knowledge: Sharing Latent Features

Latent factor models are widely used in many collaborative filtering recommender systems [36]. In these models the rating in the matrix can be represented as the product of corresponding user latent features and item latent features, which can be further explored since the latent features are similar in some cross domain scenarios.

In [146], Pan et al. proposed a method called *Coordinate System Transfer (CST)* which addresses the sparsity problem in the target domain by an adaptive approach. In particular, they performed *Singular Value Decomposition (SVD)* in the auxiliary domain that decomposes each rating matrix into the products of user latent feature factors and item latent feature factors, which are considered to be shared in the target domain. Then the transferred factors were integrated into the factorization of the rating matrix in the target domain for rebuilding the *SVD* model and generating recommendations.

In contrast to the work of [146] which learns each domain adaptively, the same authors also proposed the *Transfer by Collective Factorization (TCF)* algorithm [139] that learns all the domains collectively for handling the binary data (like/dislike). Specifically, they performed orthogonal nonnegative matrix trifactorization [147] which jointly factorizes each rating matrix in all the domains into user latent feature matrix, item latent feature matrix, and two data-dependent core matrices. Then they constructed a shared latent space with user latent feature matrices and item latent feature matrices and modeled the data-dependent effect of like/dislike by learning the core matrices.

Hu et al. [148] introduced a generalized *Cross Domain Triadic Factorization (CDTF)* model based on the TF [101]. It takes domain factors into consideration and analyzes the full triadic relation user-item-domain to reveal the user preference on items from different domains.

In the work of [149], Enrich et al. analyzed the influence of social tags in the cross-domain scenario based on the matrix factorization model [36]. Specifically, matrix factorization decompose the rating matrix in the auxiliary domain into the products of the user feature vectors and item feature vectors, in where tag factors related to an item (if avalible) are added. Then the updated item feature vectors are combined with user feature vectors (auxiliary domain) to compute rating estimations for the target domain, based on the assumption that the effect of tags on the factor model of items is cross-domains.

Fermamdes et al. [150] further explored the influence of social tags in the cross-domain recommender systems by separating user and item latent tag factors independently, for solving the scenario when a user has not assigned any tag to an item, or for items that have not been tagged yet.

Iwata et al. [151] proposed a method based on matrix factorization, assuming that latent vectors in different domains are generated from a common Gaussian distribution with a full covariance matrix. Therefore the shared latent factors can

be obtained by inferring the mean and covariance of the common Gaussian from rating matrices in different domains, which enable us to give predictions in different domains.

In summary, transferring knowledge by sharing latent features works well to reduce sparsity and increase accuracy for both auxiliary and target domains. However, it is normally computationally expensive and requires overlap of users and/or items between different domains.

2.3.2.3 Transferring Knowledge: Transferring Rating Patterns

Instead of sharing the latent features for knowledge transfer, a lot of researches focus on transferring rating patterns, based on the assumption that latent correlations may exist between preferences of group of users for group of items.

Li et al. [152] proposed an adaptive method called codebook transfer (CBT) that allows knowledge transferring from the auxiliary domain to the target domain, based on the assumption that both auxiliary and target data share the cluster-level rating patterns (codebook). The codebook is constructed by the orthogonal non-negative matrix tri-factorization [147] on the auxiliary domain, which is equivalent to the two-way K-means clustering algorithm. Then the missing ratings in the target domain can be filled by using the codebook. In this way the sparsity problem of the target domain is reduced.

Moreno et al. [153] further extended the work of [152] by proposing the *Transfer Learning for Multiple Domains (TALMUD)* approach. It extracts knowledge from multiple source domains instead of one auxiliary domain and linearly integrates the rating patterns of all source domains into one model, which proved to be more effective when multiple domains data is available.

In [154], Gao et al. introduced a *Cluster-level Latent Factor (CLF)* model to enhance the cross-domain recommendation. It integrates the common rating pattern (from the user and item clusters) [152] shared across domains with the domain-

specific rating patterns (involve the discriminative information such as topics of item clusters) in each domain, therefore generates more promising results than *CBT* method.

Transferring Rating Patterns for multi-domain recommendations was also introduced. Li et al. proposed a collective approach called rating-matrix generative model (*RMGM*) [155] that uses a probabilistic framework for effective cross-domain collaborative filtering. Unlike *CBT* that builds the codebook on a dense auxiliary domain data, *RMGM* aggregates all the rating matrices in different domains to extract the shared rating patterns. Then a probability distribution is introduced to allow users and items belong to multiple clusters, with distinct membership degrees. In this way the ratings of each domain are recovered by the expected ratings conditioned to the shared user-item clusters. *RMGM* can alleviate the sparsity problems by sharing useful knowledge across multiple related domains, which can be seen as the multi-task learning version of *CBT*.

Later, the same authors [156] further explored their work by incorporating the time factors in their proposed cross domain collaborative filtering framework [155].

Ren et al. [157] extended the work of [154] by proposing the *Probabilistic Cluster-level Latent Factor (PCLF)* model. It can be seen as the probabilistic version of *CLF* model that learns each domain simultaneously, in order to tackle the multi-task learning for all the domains.

Overall, transferring knowledge by transferring rating patterns does not need user or item overlap between domains, but it is computationally expensive.

2.4 Summary

In summary, this chapter reviews the common collaborative filtering algorithms, along with the active learning and cross-domain techniques used in the collaborative

filtering recommender systems.

Specifically, the traditional collaborative filtering algorithms are first presented, such as memory-based and model-based collaborative filtering algorithms. In Chapter 3, a new model-based collaborative filtering algorithm is proposed based on the matrix factorization models, which improves the prediction accuracy of the target recommender system. Then active learning techniques in collaborative filtering recommender systems are discussed based on the characteristics of the elicitation strategies. In Chapter 4, a general framework is proposed for applying active learning in recommender systems, for improving the performance of the whole system instead of a single user. At last, the cross-domain collaborative filtering algorithms are summarized based on how knowledge from the auxiliary domain is exploited. In Chapter 5, the existing state-of-the-art *RMGM* model is incoporated with active learning algorithm, which incurs furth improvements of the prediction accuracy of the recommender system.

Chapter 3

Matrix Factorization with Ratings Completion

3.1 Problem Statement and Motivation

A collaborative filtering recommender system usually consists of a set of users, a set of items and the preferences of users for various items, which are frequently represented as the form of [User, Item, Rating] triples. By aggregating these triples, a U-I rating matrix $\mathrm{R} \in \mathbb{R}^{m \times n}$ that consists of m users and n items can be obtained, in which each rating r_{ij} represents the preference of user i to item j. As the knowledge of preferences is very limited, the rating matrices in most recommender systems are extremely sparse. The task of collaborative filtering recommender systems is to recommend each user a list of unrated items that are ranked in a descending order based on predicted preferences (ratings). As the key point of collaborative filtering is the ratings prediction task, most algorithms transform recommending problem into the missing value estimation problem in the U-I rating matrix with high sparsity. The evaluation of the algorithms is often measured by computing the prediction accuracy of a set of unknown ratings in the rating matrix based on the predefined metrics such as *MAE* and *RMSE*.

As introduced in Section 2.1, collaborative filtering algorithms can be roughly divided into two categories: memory-based and model-based approaches. Memory-based algorithms focus on relationships between users (user-based) or items (item-based), while model-based *CF* approaches are based on prediction models that have been trained using the rating matrix. Matrix factorization methods, as one of the most successful realizations of model-based algorithms, are widely used by constructing feature matrices for users and for items, respectively. It has also shown that matrix factorization can achieve better accuracy than classic nearest neighbor methods when dealing with product recommendation [36].

In real-life scenarios, when a new user comes in, most recommender systems would only ask the user to rate a limited number of items (which is a small proportion comparing with the whole set). Therefore the rating matrices are often extremely sparse, which means there is not enough knowledge to form accurate recommendations for the user. To get precise recommendations for this user, active learning in collaborative filtering is often used to acquire more high-quality data [34] [158] [19]. However, traditional active learning methods [128] [129] [131] only evaluate each user independently and only consider the benefits of the elicitation to the 'new' user, but pay less attention to the effects of the system. In addition, in previous works [34] [128] [129], selected users were enforced to rate each elicitation through active learning process, which is hard to be true in practice. In this chapter, a *matrix completion* strategy is proposed which improves the accuracy of the whole system by automatically 'adding' more ratings for existing users. Furthermore, ratings were added one by one per request [129] or user's by user's per request [131]. The result is that the model is trained at each request, which is significantly time-consuming. In this Chapter, a series of methods is designed to obtain ratings simultaneously with matrix factorization algorithms. Through this special preprocessing step not only the computational cost is reduced, but also the performance of matrix factorization methods is greatly improved.

3.2 Matrix Factorization for Collaborative Filtering

Collaborative filtering is a very challenging work that has drawn a lot of attentions recently, as in most recommender systems rating matrices are extremely sparse. For example, the density of the famous *Netflix* [1] and *Movielens* [25] datasets is 1.18% and 6.3%, respectively, which means that only a few elements are rated. Another challenge is that the dataset used in real-world recommender systems is typically of high dimensionality. Due to high sparseness and computational complexity, directly applying traditional dimensionality reduction methods, like *SVD* algorithms, to rating matrices is not appropriate [31].

3.2.1 *Regularized SVD*

In [76], Funk proposed an effective method called *Regularized SVD* (*RSVD*) algorithm for collaborative filtering which decomposes the rating matrix into two lower rank matrices. Suppose $\mathrm{R} \in \mathbb{R}^{m \times n}$ is the rating matrix of m users and n items, $\tilde{\mathrm{R}}$ is the prediction of the rating matrix. The *Regularized SVD* algorithm finds two matrices $U \in \mathbb{R}^{k \times m}$ and $V \in \mathbb{R}^{k \times n}$ as the feature matrix of users and items:

$$\tilde{\mathrm{R}} = U^T V \tag{3.1}$$

It assumes that each user's rating is composed of the sum of preferences about various latent factors of that item. So each rating r_{ij} (corresponding prediction is represented as $\tilde{r}_{ij}$) the ith user gives to the jth item in the matrix R can be represented as:

$$\tilde{r}_{ij} = {U_i}^T V_j \tag{3.2}$$

where U_i, V_j are the feature vectors of the ith user and the jth item, respectively. Once the best approximations of U and V are obtained, the best predictions are obtained accordingly. The optimization of U and V can be performed by minimizing

the sum of squared errors between the existing scores and prediction values [76]:

$$E = \frac{1}{2} \sum_{i,j \in \kappa} (r_{ij} - \tilde{r}_{ij})^2 + \frac{k_u}{2} \sum_{i=1}^{m} {U_i}^2 + \frac{k_v}{2} \sum_{j=1}^{n} {V_j}^2 \tag{3.3}$$

where κ is a set of elements in the rating matrix R that have been assigned values, k_u and k_v are regularization coefficients to prevent over-fitting.

To solve the optimization problem in Equation (3.3), *Stochastic Gradient Descent (SGD)* is widely used and has been shown to be effective for matrix factorization [51] [77] [159]. *SGD* loops through all ratings in the training set κ and for each rating it modifies the parameters U and V in the direction of the negative gradient:

$$U_i \leftarrow U_i - \alpha \frac{\partial E_{ij}}{\partial U_i} \tag{3.4}$$

$$V_j \leftarrow V_j - \alpha \frac{\partial E_{ij}}{\partial V_j} \tag{3.5}$$

where α is the learning rate.

Unlike traditional *SVD*, *Regularized SVD* is a tool for finding those two smaller matrices, which minimize the resulting approximation error in the least square sense. By solving this optimization problem, the end result is the same as *SVD* which just gets the diagonal matrix arbitrarily rolled into the two side matrices, but could be easily extracted if needed.

3.2.2 *SVD++*

Since matrix factorization for recommender systems based on *Regularized SVD* was first proposed, several variants have been exploited with extra information on the rating matrix to improve the prediction accuracy. For example, Paterek [77] proposed an *improved Regularized SVD* algorithm by adding a user bias and an item bias in the prediction function. Koren [51] extended the *RSVD* model by consid-

ering more implicit information about rated items and proposed a *SVD++* model with the prediction function:

$$\tilde{r}_{ij} = u + \beta_i + \gamma_j + V_j^T(U_i + |I(i)|^{(-1/2)} \sum_{k \in I(i)} y_k), \tag{3.6}$$

where u is the global mean, β_i is the bias of the ith user and γ_j is the bias of the jth item. U_i is learnt from the given explicit ratings, *I(i)* is the set of items user i has provided implicit feedback for (whether each item is rated or not). $|I(i)|^{(-1/2)} \sum_{k \in I(i)} y_k$ represents the influence of implicit feedback. The implicit information enables *SVD++* to produce better performance than the *Regularized SVD* model.

3.3 The Proposed *Enhanced SVD* (*ESVD*) Model

It is important to note that the characteristics of prediction algorithms may influence the prediction accuracy. Matrix factorization methods like *Regularized SVD* and *SVD++* learn the model by fitting a limited number of existing ratings, hence the model trained with good quality as well as large quantity ratings could achieve better performance than the one with less sufficient ratings. However, in most recommendation systems, the rating matrices are extremely sparse because a user typically only rates a small proportion of items while most ratings are unknown, which motivates us to add more high quality data for matrix factorization.

3.3.1 Classic Active Learning Algorithms

Classic active learning methods focus on different individual rating elicitation strategies for a single user when a new user comes in. These strategies include:

1. *Randomization*: Items are selected randomly, which can be regarded as a baseline method (e.g., [20] [123] [131]).

2. *Popularity-based*: Items with the largest number of ratings are preferred. It is based on the assumption that the more popular the items are, the more likely that they are known by this user (e.g., [20] [21]).

3. *Entropy-based*: Items with the largest entropy are selected [20].

4. *Highest and lowest predicted*: Items with the highest or lowest predicted ratings are chosen. The items with the highest or lowest ratings are supposed to be the most liked or disliked movies for this user, which also may influence the user to rate them [123].

5. *Hybrid*: This includes $Log(popularity) * entropy$ [20], *Voting*, which consider the overall effect of previous methods [120] [131].

These strategies try to identify the most informative set of training examples, aiming to achieve better performance for users with a certain amount of ratings required from them. However, tradition active learning has several limitations:

1. First, previous works (e.g., [20] [123]) [131] focused on the accuracy of the recommendations for 'a single user', regardless of the fact that the increase of elicitations affect the performance of the whole system.

2. Furthermore, the model was trained by iterating all the users, which incurs high computational cost. With classic active learning strategies, the items selected for different users to elicit are always different. For example, the items with the highest predicted ratings for a user may not be the same as another user's since not all the users have exactly the same tastes. Hence strategy has to be applied repeatedly for each user, in order to elicit ratings which are corresponding to different items.

3. In addition, current active learning methods are based on the assumption that a user can provide ratings for any queried items, which is unrealistic and costly.

Take movie recommendation for example, to rate a movie that is generated by the active learning strategy, a user has to watch it. On the other hand, the user maybe be frustrated when asked a movie that he/she has not watched. This could lower the customer's confidence and expectation of the recommender system.

3.3.2 The Proposed Item-oriented Approach

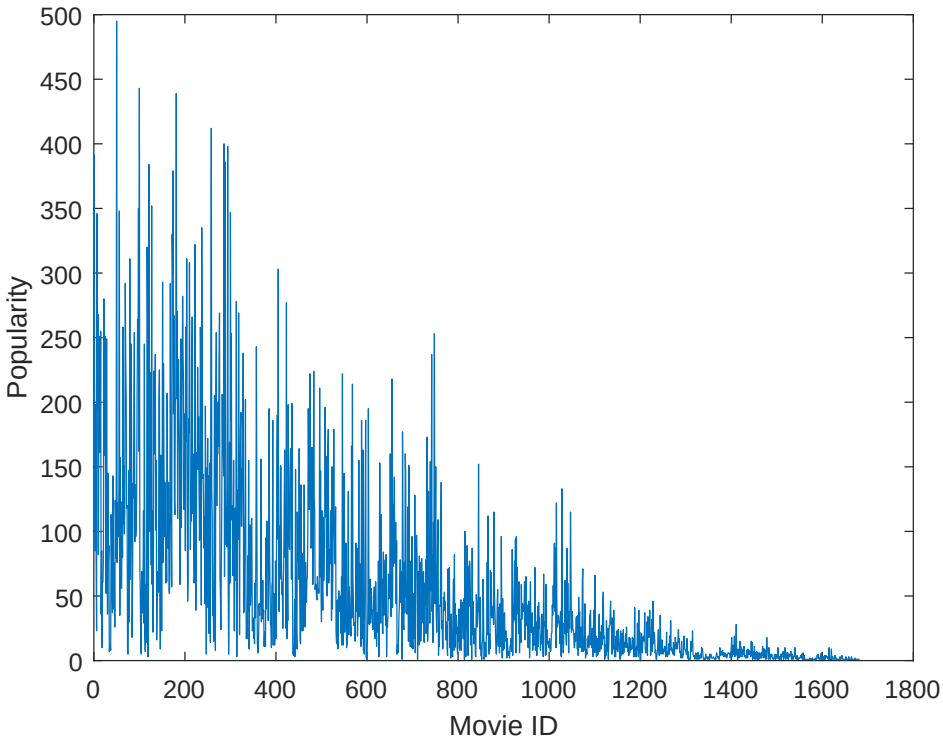


Figure 3.1: The number of ratings each item has received (popularity) in *Movielens 100K*

From Figure 3.1 it can be observed that the movie popularity may vary significantly. Take the *Movielens 100K* dataset for example, the maximal and minimum level of popularity is 495 and 0, respectively, which means that the most popular movie is rated by 495 users. Popularity is based on the number of ratings regarding to each item only which is irrelevant to users, therefore the popularity of each movie remains

the same for all the users. In Algorithm 3.1, by selecting N most popular movie for all the users a new sub-matrix could be obtained (as shown in Figure 3.2), based on the idea that users tend to rate world-famous movies than the less known movies. Then the missing values in this sub-matrix would be the desirable movies in some sense for the users who missed before. Unlike traditional active learning that queries only new users for a certain number of ratings in each iteration, the proposed strategy predicts these specific ratings for all the users at the same time in one iteration based on matrix factorization algorithms on this sub-matrix. After adding these ratings to the original rating matrix, a more accurate matrix factorization model could be trained.

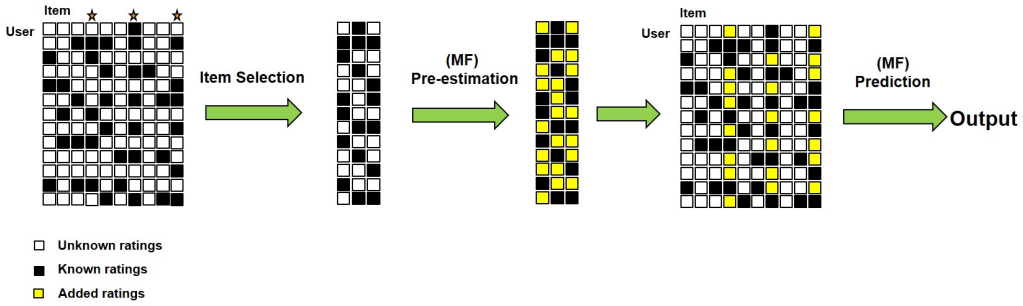


Figure 3.2: Procedures of Item-oriented Approach

In summary, this item-oriented (based on item popularity) approach pre-estimate ratings of only popular movies for all the users simultaneously (in contrast to active learning that elicit ratings for each user iteratively), in order to improve the performance of the whole system. Therefore, it reduces the training time of the matrix factorization model from as high as the number of users (for active learning) to only 2 (the proposed method), which saves a lot of computational cost.

3.3.3 The Proposed User-oriented Approach

In contrast to traditional active learning for collaborative filtering which selects a number of items to rate so as to improve the rating prediction for the user, Carenini et al. [21] proposed an alternative active learning method that elicits ratings by

Algorithm 3.1 The Proposed Item-oriented Approach

Input: Rating matrix $\mathrm{R} \in \mathbb{R}^{m \times n}$, where $Q_{j \in [1,n]} \in \mathbb{R}^{m \times 1}$ is the column vector, κ is a set of elements in the rating matrix that have been assigned values; the number of items selected in the sub-matrix based on popularity N;

Output: RMSE of the test set;

Step 1: Sort items based on popularity in the descending order $j(1), j(2), ..., j(m)$;

Step 2: Create a sub-matrix M_1 by selecting the top N items (columns) of R based on the popularity. Therefore $M_1 = [Q_{j(1)}, Q_{j(2)}, ..., Q_{j(N)}](N < m)$;

Step 3: Apply basic matrix factorization (*Regularized SVD*) on matrix M_1 to obtain feature matrices U and V according to Equation (3.1);

Step 4: Predict every missing value in sub-matrix M_1 to acquire a non-null matrix M_1' according to Equation (3.2). Then a series of ratings L_1 is obtained, such that $L_1 = \{$

$r_{i_{k(1)},j(1)}, r_{i_{k(2)},j(1)}, ..., r_{i_{k(n)},j(1)},$

$r_{i_{k(1)},j(2)}, r_{i_{k(2)},j(2)}, ..., r_{i_{k(n')},j(2)},$

.....,

$r_{i_{k(1)},j(N)}, r_{i_{k(2)},j(N)}, ..., r_{i_{k(n'')},j(N)}\}$

where $r_{i_k,j} \notin \kappa$;

Step 5: Fill ratings in the original matrix R with every predicted value by **Step 4** to acquire a new rating matrix R'. That means the extra ratings are added into the training set $\kappa = \{\kappa, L_1\}$;

Step 6: Apply basic matrix factorization (*Regularized SVD*) on matrix R' to obtain feature matrices *U'* and *V'* according to Equation (3.1)l. Then predict the target ratings (test set) according to Equation (3.2) and calculate *RMSE* according to Euqation (1.2);

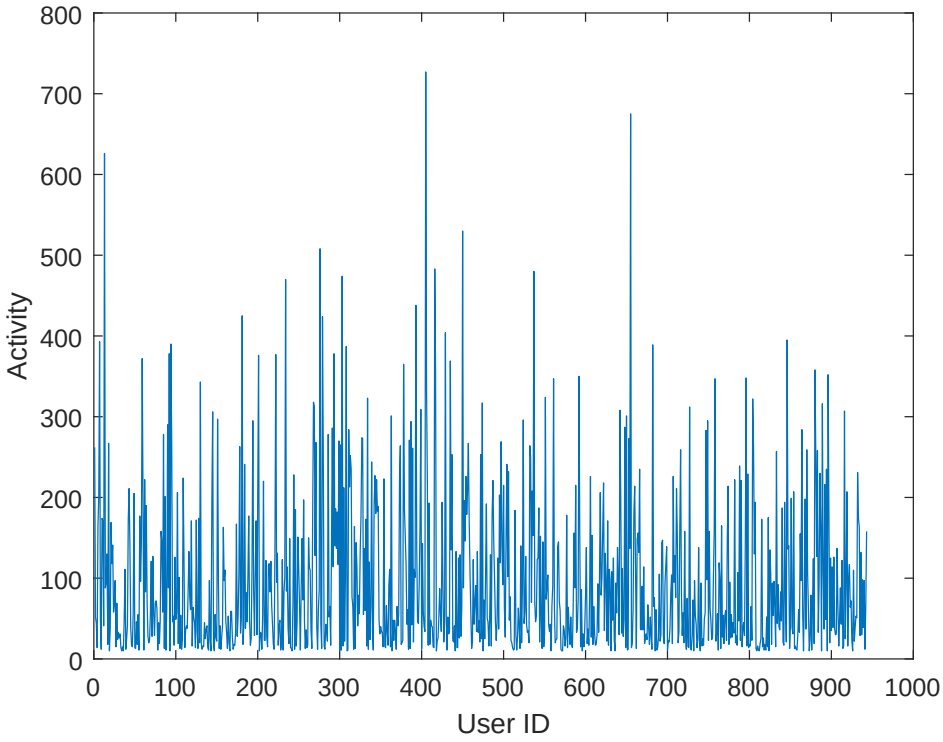


Figure 3.3: The number of ratings each user has rated (activity) in *Movielens 100K*

choosing some special users to rate a specific item in order to improve the rating prediction for the item. Likewise, a user-oriented approach is also proposed to further explore the potential of the proposed method.

Generally, the number of movies each user has rated varies significantly as shown in Figure 3.3 (e.g., in the *Movielens 100K* dataset the maximal and minimum number for different user's are 727 and 10, respectively). Though active users who are enthusiastic about movies may watch far more than the ones who are not into movies, there still exist some movies the users have watched but not yet rated. Therefore it is easier to accept that active users have high possibility to give ratings to their unrated movies, but little chance for the users who had no interest in providing ratings before (with a small number of ratings in the data set). Therefore the user-oriented approach is proposed (Algorithm 3.2) by selecting this kind of

Algorithm 3.2 The Proposed User-oriented Approach

Input: Rating matrix R $\in \mathbb{R}^{m \times n}$, where $P_{i \in [1,m]} \in \mathbb{R}^{1 \times n}$ is the row vector, κ is a set of elements in the rating matrix that have been signed values; the number of users selected in the sub-matrix based on activity N';

Output: RMSE of the test set;

Step 1: Sort users based on activity in descending order $i(1), i(2), ..., i(n)$;

Step 2: Create a sub-matrix M_2 by selecting the top N users (rows) of R based on the activity. Therefore $M_2 = [P_{i(1)}, P_{i(2)}, ..., P_{i(N')}](N' < n)$;

Step 3: Apply basic matrix factorization (*Regularized SVD*) on matrix M_2 to obtain feature matrices U and V according to Equation (3.1);

Step 4: Predict every missing value in sub-matrix M_2 to acquire a non-null matrix M_2' according to Equation (3.2). Then a series of ratings L_2 is obtained, such that $L_2 = \{$

$r_{i(1),j_k(1)}, r_{i(1),j_k(2)}, ..., r_{i(1),j_k(n)},$
$r_{i(2),j_k(1)}, r_{i(2),j_k(2)}, ..., r_{i(2),j_k(n')}$
$r_{i(N'),j_k(1)}, r_{i(N'),j_k(2)}, ..., r_{i(N'),j_k(n'')}\}$
where $r_{i,j_k} \notin \kappa$;

Step 5: Fill ratings in the original matrix R with every predicted value by **Step 4** to acquire a new rating matrix R$'$. That means the extra ratings are added into the training set $\kappa = \{\kappa, L_2\}$;

Step 6: Apply basic matrix factorization (*Regularized SVD*) on matrix R$'$ to obtain feature matrices U' and V' according to Equation (3.1); Then predict the target ratings (test set) according to Equation (3.2) and calculate *RMSE* according to Euqation (1.2);

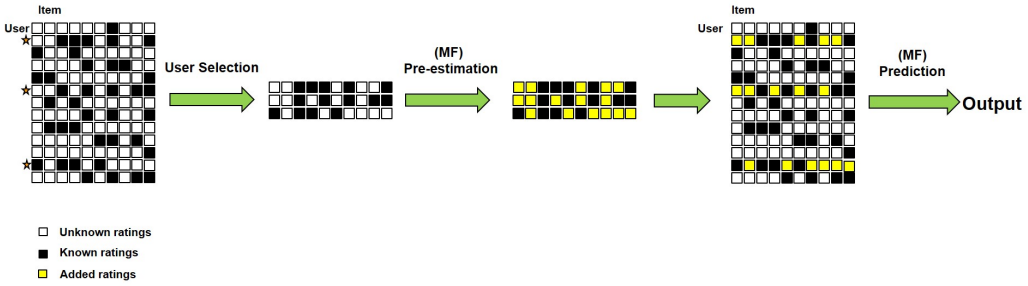


Figure 3.4: Procedures of User-oriented Approach

special users based on the number of movies they have rated. After these movie enthusiasts are chosen (as shown in Figure 3.4), ratings of the movies they never rate (as the missing values in the new sub-matrix) would be predicted by matrix factorization algorithms. Then these new ratings are added to the original matrix for generating better recommendations.

In brief, this user-oriented (based on user activity) approach tries to improve the performance of the whole system by pre-estimating ratings simultaneously of all movies for only active users. Therefore it also has the benefits that item-oriented approach has. However, both algorithms may still incur significant computational cost and distortion of the original model because of the extensive selection of added ratings, especially when the number of popular movies or active users selected in the sub-matrix is large.

3.3.3.1 The Proposed *ESVD* (Density-Oriented Approach)

So far an item-oriented approach and a user-oriented approach are presented, both based on the idea that pre-estimating a group of reliable and meaningful ratings simultaneously for the matrix factorization model to learn. The reason why these new ratings are reliable is because they are predicted from the denser sub-matrix, which consists of the largest number of ratings from either the item-view or the user-view by matrix factorization algorithms. The recommender system with sufficient ratings could easily generate accurate recommendations. Typically the denser the

Algorithm 3.3 The Proposed *ESVD* (Density-Oriented Approach)

Input: Rating matrix R $\in \mathbb{R}^{m \times n}$, where $P_{i \in [1,m]} \in \mathbb{R}^{1 \times n}$ is the row vector and $Q_{j \in [1,n]} \in \mathbb{R}^{m \times 1}$ is the column vector, κ is a set of elements in the rating matrix that have been assigned values; The number of items selected in the sub-matrix based on popularity N and the number of users selected in the sub-matrix based on activity N';

Output: *RMSE* of the test set;

Step 1: Sort both items and users in the descending order based on popularity and activity respectively. $j(1), j(2), ..., j(m)$; $i(1), i(2), ..., i(n)$;

Step 2: Create a sub-matrix M_1 by selecting the top N items (columns) of R based on the popularity. Therefore $M_1 = [Q_{j(1)}, Q_{j(2)}, ..., Q_{j(N)}](N < m)$;
And also create a sub-matrix M_2 by selecting the top N' users (rows) of R based on the activity. Therefore $M_2 = [P_{i(1)}, P_{i(2)}, ..., P_{i(N')}](N' < n)$;

Step 3: Create a sub-matrix M_3 by selecting the intersection of top N items (columns) and top N' users (rows) based on the popularity and activity. Therefore $M_3 = M_1 \bigcap M_2$;

Step 4: Apply basic matrix factorization (*Regularized SVD*) on matrix M_3 to obtain feature matrices U and V according to Equation (1). Then predict every missing value in sub-matrix M_3 to acquire a non-null matrix M'_3 according to Equation (2). Then a series of ratings L is obtained, such that $L = \{r_{i_{k(1)}, j_{t(1)}}, ..., r_{i_{k(n)}, j_{t(n')}}\}$ where $r_{i_k, j_t} \in (M_3 \bigcap \neg\kappa)$;

Step 5: Fill ratings in the original matrix R with every predicted value by **Step 4** to acquire a new rating matrix R$'$. That means the extra ratings are added into the training set $\kappa = \{\kappa, L\}$;

Step 6: Apply basic matrix factorization (*Regularized SVD*) on matrix R$'$ to obtain feature matrices *U'* and *V'* according to Equation (3.1). Then predict the target ratings (test set) according to Equation (3.2) and calculate *RMSE* according to Euqation (1.2);

matrix is, the better the matrix factorization model is obtained. Take the *Movielens 100K* dataset as an example, the density of the original matrix is 6.3%. However, if only 5% of the most popular movies are chosen, a sub-matrix obtained of density 29.47% which consists of more ratings that have been already rated by the users. While selecting the 5% of the most active users, the density of the new sub-matrix obtained is 23.33%. Based on this observation a density-oriented approach is proposed which combines previous item-oriented and user-oriented methods in Algorithm 3.3.

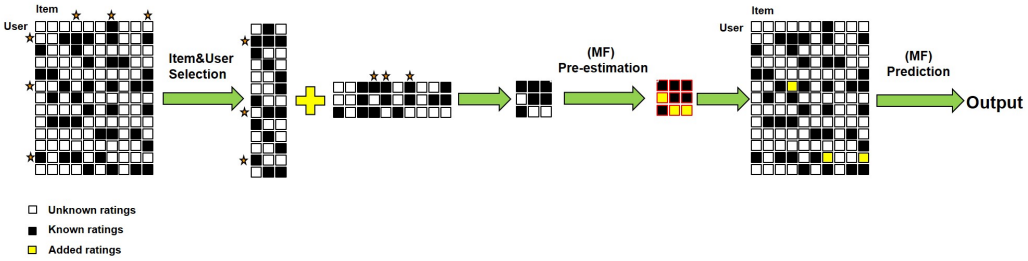


Figure 3.5: Procedures of *ESVD*

ESVD is based on the assumption that the recommender system was first built with a set of the most popular movies that are rated by a set of the most active users. Because both the popularity of items and the activity of users depend on the numbers of ratings each user rates or each movie is rated, by choosing the most N popular items (columns) and the most N' active users (rows) the densest sub-matrix is obtained (as shown in Figure 3.5). For example with *Movielens 100K* dataset, if choose 5% of the most popular movies and most active users, the density of the newly-formed sub-matrix would be 77.28% (Step 3 in Algorithm 3.3). The missing values in this sub-matrix can be explained as ratings of the most famous movies but have not been rated by a group of the most active users. Therefore the recommendations generated by this recommender system should be of high accuracy. Afterwards some rare movies most people probably have not seen and users with very few ratings are added into the dataset (the orginal matrix), which could lower

the prediction accuracy of the whole system. To achieve better performance, the ratings (pre-estimations) generated from the former recommender system could be used (by applying matrix factorization on the sub-matrix) as the known knowledge for further learning and inference. Finally a more accurate matrix factorization model can be learnt by fitting the existing ratings and extra high quality ratings.

3.3.4 Evaluation

3.3.4.1 Datasets and Experimental Setup

Experiments of the proposed item-oriented, user-oriented and density-oriented approach (*ESVD*) are conducted on the classic recommender system datasets: the *Movielens 100K* and the subset of the *Netflix* (the first 106,150 ratings are extracted from the full *Netflix* dataset as the subset of *Netflix*, which are made by 1,910 users on 1,780 movies). Some experiments with the larger version are also performed and obtained similar results. However, it requires much longer time to perform the experiments since the models are trained and tested each time for different choice of N and N'. Therefore, the smaller datasets *Movielens 100K* and subset of original *Netflix* are focused to be able to run more experiments, in order to explore how these two parameters affects the results of the proposed matrix factorization methods.

Normally each dataset is partitioned into a training set and a test set. The model is trained on the training set and the quality of results is usually measured by the *Root Mean Square Error* (*RMSE*) of the test set. *RMSE* is used as the default metric, which is widely used in the *Netfilx* Competition [1] and proved to be effective for measuring recommender systems.

The number of the latent factors (rank) k are set to be 10 for training each matrix factorization model. Although increasing it does raise the performance, the computational cost is proportional to latent factors. For matrix factorization of the sub-matrix, the coefficient of the regularization term k_u and k_v are 0.01 and 0.05 for the *Movielens 100K* and *Netflix* datasets, respectively. And the learning rate

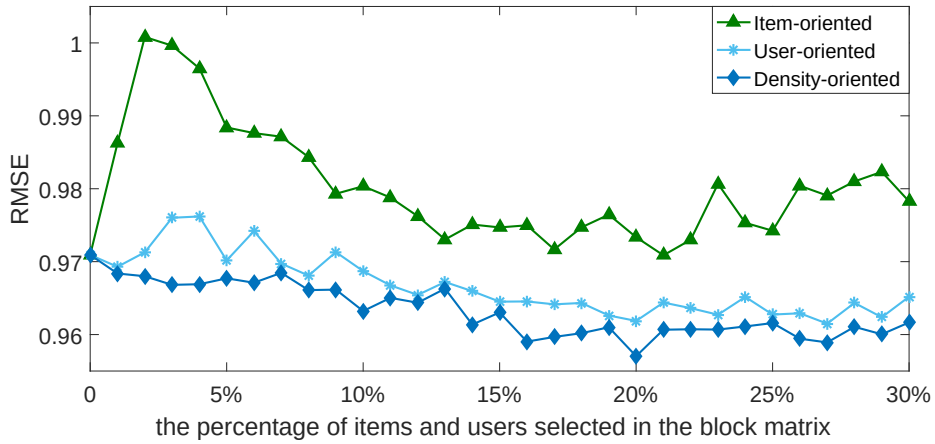


Figure 3.6: *Movielens*: *RMSE* comparisons of proposed methods based on *SVD*

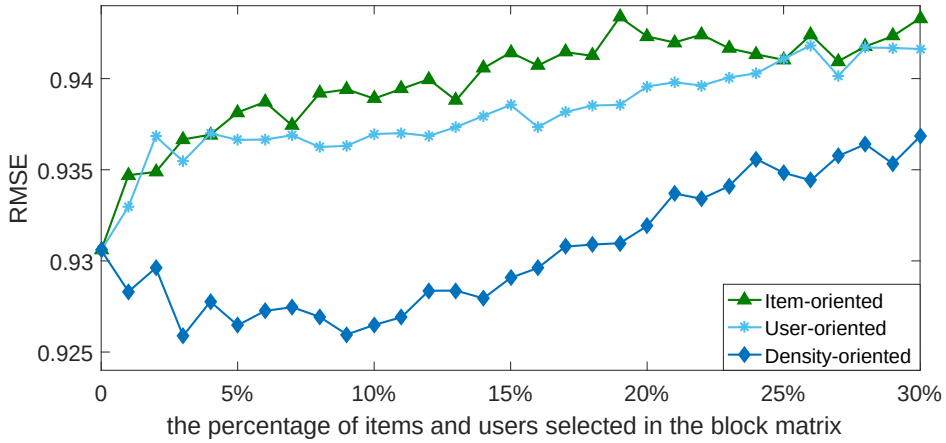


Figure 3.7: *Netflix*: *RMSE* comparisons of proposed methods based on *SVD*

α is 0.1 with a decrease by a factor of 0.9 each iteration for both datasets. For matrix factorization of the rating matrix R$'$ (with pre-estimations), the coefficient of regularization term $k_u{}'$ and $k_v{}'$ are 0.1 for both datasets, and the learning rate α is 0.01 and 0.05 with decrease by a factor of 0.9 each iteration for the *Movielens 100K* and *Netflix* datasets, respectively.

Table 3.1: *RMSE* of *ESVD* on *Movielens 100K* (The Density-Oriented Approach)

Items&Users	Block Density	Extra Ratings	*RMSE*
0%	null	null	0.9709
5%	77.28%	897	0.9677
10%	65.20%	5496	0.9632
15%	53.90%	16381	0.9630
20%	45.66%	34508	0.9570

Table 3.2: *RMSE* of *ESVD* on *Netflix* (The Density-Oriented Approach)

Items&Users	Block Density	Extra Ratings	RMSE
0%	null	null	0.9306
5%	59.06%	3498	0.9265
10%	43.59%	19179	0.9265
15%	33.10%	51268	0.9291
20%	25.51%	101298	0.9319

3.3.4.2 Experimental Results

Figure 3.6 and Figure 3.7 show the results of the proposed methods based on how many items and users selected (simply setting $N = N'$ in this case) in the sub-matrix on the *Movielens 100K* and *Netflix* datasets, respectively. All the methods start at 0 point where no extra filling is added into the learning process, which is the same as *RSVD*. It can be seen that the results of item-oriented approach and user-oriented sometimes are not promising. Because in the item-oriented (or user-oriented) approach only pre-estimations are added based on the most popular movies (or users), which may lead to a lot of bias and distort the latent factor model. For example, most people prefer happy endings, and the consequence is that comedies are more popular than tragedies. As a result, a lot of comedy movies would be elicited for each user to give ratings which leads to more weights on the factor corresponding to comedies in the latent factor model (*RSVD* in this case). It is apparent from the Figure 3.6 and Figure 3.7 that the proposed *ESVD* consistently outperforms other methods including the baseline method: *RSVD*.

In Table 3.1 and Table 3.2, the experimental results of the proposed density-

oriented (*ESVD*) method are illustrated which incorporates both item-oriented and user-oriented approach on the *Movielens 100K* and *Netflix* datasets. Different *RMSE* are compared based on how many items and users ($N = N'$ from 0% to 20%) selected. Note that the basic matrix factorization is a special case of the proposed method when setting $N = 0\%$, which is used as the baseline for comparision. After selecting a certain percentage of items and users, a sub-matrix is formed. It can be observed that the more items and users are chosen, the much sparser the sub-matrix is. The missing values in the sub-matrix are chosen to be pre-estimated ratings. Although sparser matrix may lead to a less accurate matrix factorization model and the quality of pre-estimations may not as good as the ones from the denser matrix, the number is increased. Therefore more ratings can be obtained and put into the process of learning the target matrix factorization model. At last predictions are computed on the test set and corresponding results are obtained. Because the sub-matrix is the intersection of the largest N items and N' users, its density is much greater than the one from item-oriented or user-oriented approach. Even with fewer ratings to be added compared with item-oriented and user-oriented, the results are better.

In the experiments, it can be observed that for the *Movielens 100K* dataset the performance fluctuates as the number of projects increases (Figure 3.6). While for the *Netflix* dataset (Figure 3.7), the performance drops at first then it deteriorates (the lower *RMSE* the better performance) as N goes up. This is mainly because the *Netflix* dataset is much sparser than the *Movielens 100K* dataset. While adopting the *ESVD* algorithm, as N increases, more poor quality data is added into the learning process and leads to the distortion of the model (Figure 3.7). The optimal point (N) that balances the quality (density of sub-matrix) and the quantity (number of added ratings) depends on the distribution of ratings. For the *Movielens 100K* dataset, the proposed *ESVD* can reach 0.9570 (when $N = 20\%$) which reduces the *RMSE* by 0.0139 compared with the *Regularized SVD* 0.9709. For the *Netflix*

dataset, it could lower the *RMSE* by 0.0047 (from 0.9306 to 0.9259 when $N = 3\%$).

3.3.5 The Proposed *ESVD++*

Broadly speaking, the proposed *ESVD* approach can be seen as a preprocessing step and it can be incorporated with other variants of SVD models, such as *SVD++* [51] to form a new approach called *ESVD++*. *ESVD++* is conducted by just changing the prediction algorithm from *SVD* to *SVD++*. Compared with the *SVD* model, *SVD++* improves the prediction accuracy by adding biases and the implicit information $I(i)$, and the prediction function is shown in Equation (3.6). Specifically, $I(i)$ contains all the items for which the ith user has provided a rating, even if the value is unknown. Therefore, for prediction of added ratings as shown in Step 4 of Algorithm 3.3, $I(i)$ is set to be the number of existing ratings and the missing values in the sub-matrix that are also shown in the test set. For prediction of the test set as shown in Step 6 of Algorithm 3.3, $I(i)$ is the same as the one in original matrix without considering extra ratings.

As the strategy is the same as *ESVD*, the ratings that need to be elicited are also the same. Here the process of searching for the optimal value for N is skipped and the results are listed directly. The *ESVD++* outperforms the state-of-art *SVD++* model and greatly reduces the *RMSE* by 0.0214 (from the baseline *SVD++* 0.9601 to 0.9387 when $N = 10\%$) and 0.004 (from the baseline *SVD++* 0.9222 to 0.9182 when $N = 8\%$) for the *Movielens 100K* and *Netflix* datasets, respectively.

3.4 The Proposed *Multilayer ESVD* (*MESVD*)

In the *ESVD* procedure, all the extra ratings are predicted in a single matrix factorization model simultaneously, which could lead to a lot of bias and distort the original model when the number of pre-estimations is large. To alleviate this prob-

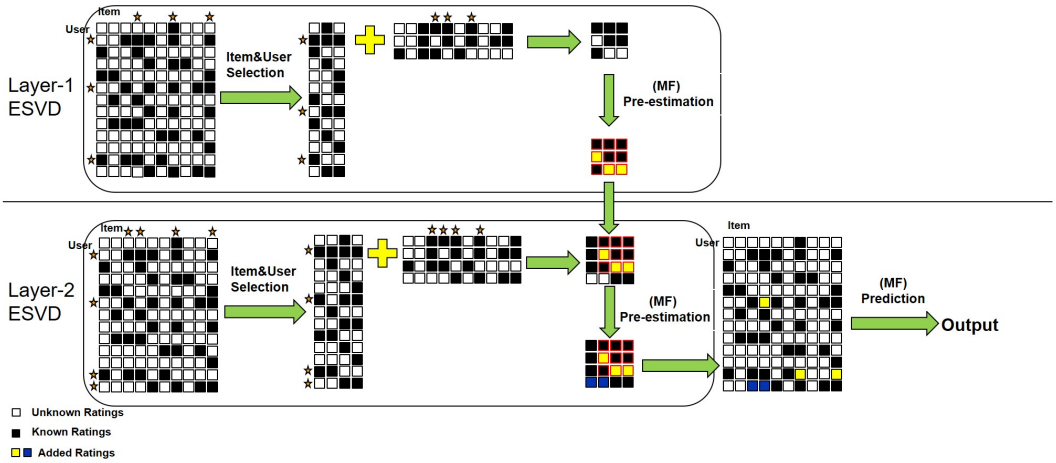


Figure 3.8: Procedures of *Multilayer ESVD*

lem a method called *Multilayer ESVD* (*MESVD*) is proposed in Algorithm 3.4 which obtains the fillings incrementally through multiple matrix factorization on different sub-matrices.

The example of the *Two-layer ESVD* is shown in Figure 3.8. First a set of sub-matrices are created in each layer by selecting the intersections of different numbers of columns and rows (as stared) based on the number of ratings each item or each user has, respectively. Therefore each smaller sub-matrix (with red frame) in the upper layer can be seen as a part of the bigger sub-matrix in the lower layer. The missing values (yellow ratings) in the smaller sub-matrix can be predicted by the matrix factorization method and then they would be regarded as the known ratings in the bigger sub-matrix. Similar to deep learning, the outputs generated by each upper layer are utilized as the inputs of each lower layer, for enhancing the prediction accuracy of their outputs (pre-estimations) which could be reused as the inputs of next lower layer. For example in Figure 3.8 ratings in black and yellow are known in the layer-2, therefore the sub-matrix in the next layer is much denser than the one without pre-estimations (ratings in yellow) from upper layer. In this way fillings are predicted iteratively layer by layer. At last all the pre-estimated ratings are added into the original matrix to evaluate the performance of the whole system.

Basically, the *MESVD* approach is based on the assumption that the recommender system was built by a very dense matrix with sufficient ratings at first. Therefore the recommendations (represented as missing values in the sub-matrix) were reliable and can be regarded as the known knowledge. After that it is better to keep inviting the most active users to rate the most popular movies for the recommender system than the one in the minority. In this way, each time a set of movies and users are added in the system, iteratively generating knowledge for further learning and inference (from the upper layer to lower layer). As a result, better performance can be obtained by learning the current systems with extra knowledge generated in each of the sub-system's layer.

3.4.1 Experimental Results

Experiments of the *MESVD* method on the *Movielens 100K* and *Netflix* datasets are also conducted. The corresponding results are shown in Table 3.3 and Table 3.4. For the *Movielens 100K* dataset experiments of *ESVD* are conducted when *N=20%* (optimal point), *Two-layers ESVD* where the first layer is 10% and the 2nd layer is 20%, *Four-layers ESVD* with layers from 5% to 20% with 5% interval (setting *N=N'*). Specifically, in the first experiment ratings are elicited from the sub-matrix of density 45.66%; In *Two-layers ESVD*, the first 5496 ratings are elicited from the sub-matrix of density 65.20% while the rest are elicited from the sub-matrix of density 54.32%; In *Four-layers ESVD* ratings are elicited layer by layer for four times, each time ratings are elicited from the much denser matrix. It can be seen that as the result the numbers of fillings in total are the same, as the added sub-matrices in the ending layers are the same. The performance gets better from single layer to Four-layers, for the reason that the quality of extra ratings gets better.

For the *Netflix* dataset four experiments are performed: *ESVD* when *N=10%*, *Two-layers ESVD* where the first layer is 5% and the 2nd layer is 10%, *Four-layers ESVD* with layers from 2.5% to 10% with 2.5% interval, and *Six-layers ESVD* with

Algorithm 3.4 The Proposed *Multilayer ESVD* (*MESVD*)

Input: Rating matrix R $\in \mathbb{R}^{m\times n}$, where $P_{i\in[1,m]} \in \mathbb{R}^{1\times n}$ is the row vector and $Q_{j\in[1,n]} \in \mathbb{R}^{m\times 1}$ is the column vector, κ is a set of elements in the rating matrix that have been assigned values; The total number of layers $x \in [1, min(m,n)]$. The numbers of items selected in the sub-matrix based on popularity $N_1 < N_2 < ... < N_x \in [1, m]$ and the numbers of users selected in the sub-matrix based on activity $N'_1 < N'_2 < ... < N'_x \in [1, n]$;

Output: RMSE of the test set;

Step 1: Sort both items and users in descending order based on popularity and activity respectively. $j(1), j(2), ..., j(m)$; $i(1), i(2), ..., i(n)$;

Step 2: Create a series of sub-matrices $M_{1(1)}, M_{1(2)}, ..., M_{1(x)}$ by selecting different numbers of top $N_1, N_2, ..., N_x$ items (columns) of R based on the popularity. Therefore each sub-matrix $M_{1(d)} = [Q_{j(1)}, Q_{j(2)}, ..., Q_{j(N_d)}](d \in [1, x])$;

Step 3: Create a series of sub-matrices $M_{2(1)}, M_{2(2)}, ..., M_{2(x)}$ by selecting different numbers of top $N'_1, N'_2, ..., N'_x$ of users (rows) of R based on the activity. Therefore each sub-matrix $M_{2(d)} = [P_{i(1)}, P_{i(2)}, ..., P_{i(N'_d)}](d \in [1, x])$;

Step 4: Create a series of sub-matrices $M_{3(1)}, M_{3(2)}, ..., M_{3(x)}$ by selecting the intersection of top N items (columns) and top N' users (rows) of R based on the popularity and activity. Therefore each sub-matrix $M_{3(d)} = M_{1(d)} \bigcap M_{2(d)}(d \in [1, x])$;

For $(s = 0; s < x; s{+}{+})$

{

Step 5: Apply basic matrix factorization (*Regularized SVD*) on matrix $M_{3(1+s)}$ to obtain feature matrices U and V according to Equation (3.1); Then predict every missing value in sub-matrix $M_{3(1+s)}$ to acquire a non-null matrix $M'_{3(1+s)}$ according to Equation (3.2). Then a series of ratings $L_{3(1+s)}$ is obtained, such that $L_{3(1+s)} = \{r_{i_{k(1)},j_{t(1)}}, ..., r_{i_{k(n)},j_{t(n')}}\}$where $r_{i_k,j_t} \in (M_{3(1+s)} \bigcap \neg\kappa)$;

Step 6: Fill ratings in the original matrix R with every predicted value by **Step 5** to acquire a new rating matrix R$'$. That means the extra ratings are added into the set of existing ratings. $\kappa = \{\kappa, L_{3(1+s)}\}$;

}

Step 7: Apply basic matrix factorization (*Regularized SVD*) on matrix R$'$ to obtain feature matrices U' and V' according to Equation (3.1); Then predict the target ratings (test set) according to Equation (3.2) and calculate *RMSE* according to Euqation (1.2);

Table 3.3: *RMSE* of *MESVD* on *Movielens 100K*

	Item&User	Block Density	Extra Ratings	*RMSE*
RSVD	N=0	0	0	0.9709
ESVD	N=20%	45.66%	34508	0.9570
Two-layers ESVD	N=[10%,	65.20%	5496	
	20%]	54.32%	29012	0.9564
Four-layers ESVD	N=[5%,	77.28%	897	
	10%,	70.88%	4599	
	15%,	69.37%	10885	
	20%]	71.46%	18127	0.9561

layers from 5% to 10% with 1% interval. It can be observed that *Two-layers ESVD* yields better performance than *ESVD*, because each batch of fillings are predicted from the denser matrices with better accuracy. For the same reason, better results can be obtained based on *Four-layers ESVD* than *Two-layers ESVD*. When compared *Six-layers ESVD* with *Two-layers ESVD*, the first batch of fillings are the same, however, the rest are of better quality because they are learnt layer by layer in the denser matrices. When compared *Six-layers ESVD* with *Four-layers ESVD*, although all the fillings are learnt by more iterative times, the first batch of extra ratings are of poorer quality. As a result, the result of *Six-layers ESVD* is not as good as *Four-layers ESVD*. In summary, although the optimal point of N is not selected, better performance is obtained than *ESVD* (0.9259 when $N = 3\%$).

Experimental results show that the quality of *MESVD* depends on the number of layers and the choice of each layer, which still remain further study. In *ESVD* algorithm, decent result cannot be obtained if the number of items and users selected in the sub-matrix N is inappropriate. Through *MESVD* method, this problem can be alleviated with comparable or better results. With optimal point of N, better performance can still be obtained by learning the added ratings iteratively through *MESVD* method. The improvements of *MESVD* approach is limited, as the ratings added in the original matrix are the same when compared with *ESVD* approach. However, if the training time is not the priority concern, *MESVD* (the iteration of

Table 3.4: *RMSE* of *MESVD* on *Netflix*

	Item&User	Block Density	Extra Ratings	*RMSE*
RSVD	N=0	0	0	0.9306
ESVD	N=10%	43.59%	19179	0.9265
Two-layers ESVD	N=[5%,	59.06%	3498	
	10%]	53.88%	15681	0.9262
Four-layers ESVD	N=[2.5%,	67.04%	712	
	5%,	67.39%	2786	
	7.5%,	68.33%	6068	
	10%]	71.72%	9613	0.9248
Six-layers ESVD	*N*=[5%,	59.06%	3498	
	6%,	83.76%	1998	
	7%,	84.35%	2621	
	8%,	86.11%	3017	
	9%,	86.74%	3650	
	10%]	87.07%	4395	0.9255

training matrix factorization model depends on the number of layers) is preferable.

3.5 The Proposed Extensions of *ESVD*

So far *ESVD* has been presented which applies *SVD* with ratings completion strategy that best approximates a given matrix with missing values. Experimental results show that the extra fillings do improve the performance of the system. The reason is that the model is learnt by extra high quality ratings that are predicted from the dense sub-matrix based on item popularity and user activity. Based on this theory two extensions are proposed in order to acquire better fillings for different kinds of datasets.

3.5.1 The Proposed Item-wise *ESVD* (*IESVD*)

When dealing with the rating matrix of which the number of users is far greater than the number of items, each item has been rated by a large number of users (popularity) but each user only rate few items (activity) in average. Therefore, popular items have more impacts than active users on the density of newly-formed

Algorithm 3.5 The Proposed *Item-wise ESVD* (*IESVD*)

Input: Rating matrix $\mathrm{R} \in \mathbb{R}^{m \times n}$, where $Q_{j \in [1,n]} \in \mathbb{R}^{m \times 1}$ is the column vector, κ is a set of elements in the rating matrix that have been assigned values; The number of items selected in the sub-matrix based on popularity N and the number of users selected in the sub-matrix based on activity N';

Output: RMSE of the test set;

Step 1: Sort items in the descending order based on popularity $j(1), j(2), ..., j(m)$;

Step 2: Create a sub-matrix M_1 by selecting the top N items (columns) of R based on the popularity. Therefore $M_1 = [Q_{j(1)}, Q_{j(2)}, ..., Q_{j(N)}](N < m)$ where $P_{i \in [1,m]} \in \mathbb{R}^{1 \times N}$ is the row vector of M_1;

Step 3: Sort users based on activity of the sub-matrix M_1 in descending order $i(1), i(2), ..., i(n)$;

Step 4: Create a sub-matrix M_2 by selecting the top N' users (rows) of M_1 based on the activity. Therefore $M_2 = [P_{i(1)}, P_{i(2)}, ..., P_{i(N')}](N' < n)$;

Step 5: Apply basic matrix factorization (*Regularized SVD*) on matrix M_2 to obtain feature matrices U and V according to Equation (3.1); Then predict every missing value in sub-matrix M_2 to acquire a non-null matrix M_2' according to Equation (3.2). Then a series of ratings L is obtained, such that $L = \{r_{i_{k(1)}, j_{t(1)}}, ..., r_{i_{k(n)}, j_{t(n')}}\}$ where $r_{i_k, j_t} \in (M_2 \bigcap \neg \kappa)$;

Step 6: Fill ratings in the original matrix R with every predicted value by **Step 5** to acquire a new rating matrix R$'$. That means the extra ratings are added into the set of existing ratings. $\kappa = \{\kappa, L\}$;

Step 7: Apply basic matrix factorization (*Regularized SVD*) on matrix R$'$ to obtain feature matrices *U'* and *V'* according to Equation (3.1). Then predict the target ratings (test set) according to Equation (3.2) and calculate *RMSE* according to Equation (1.2);

sub-matrix. As a result, obtaining sub-matrix based on item popularity and user activity simultaneously is not appropriate under such circumstance.

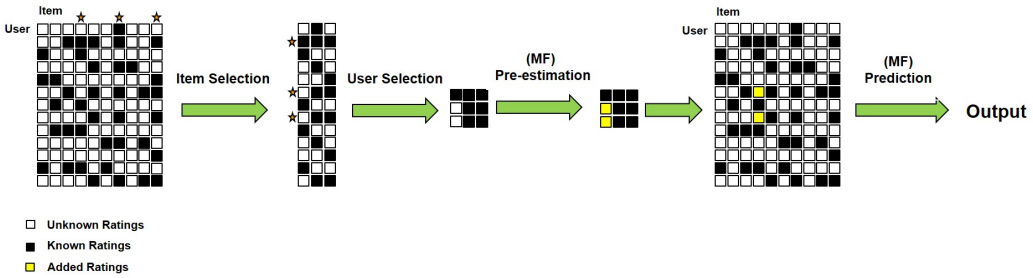


Figure 3.9: Procedures of Item-wise *ESVD*

The *Item-wise ESVD (IESVD)* (Algorithm 3.5) is proposed by first selecting a number of the most popular items to form a sub-matrix as the *ESVD* does (step 1 of Figure 3.9). Then only the active users that have seen these specific movies (stared in step 2) are chosen. This means users are selected based on the number of ratings in the sub-matrix only instead of the whole rating matrix. In this way a denser sub-matrix can be obtained than the one from the *ESVD* method. Likewise, the missing values in the sub-matrix can be pre-estimated by matrix factorization method. Finally, the predicted ratings are filled in the original matrix. Therefore the new matrix factorization model is learnt and tested based on the newly-formed rating matrix.

3.5.2 The Proposed *User-wise ESVD*

Likewise, in the datasets that consists of much more items than users, the quantity of ratings each user rate (activity) is much greater than the quantity of ratings each items is rated (popularity) in average. Therefore the *User-wise ESVD* (*UESVD*) (Algorithm 3.3) is proposed as shown in Figure 3.10. Initially, a number of the most active users are selected to form a sub-matrix based on the number of ratings each user has rated. Then the most popular items that the active users have seen are chosen to form the sub-matrix, i.e. the items with most ratings in the sub-matrix only. As the result a denser sub-matrix is obtained than the one from *ESVD*. The rest procedures are the same as the *ESVD* algorithm.

Therefore both *IESVD* and *UESVD* train the matrix factorization model twice by automatically adding pre-estimations in the data set. However, the *IESVD* and *UESVD* approaches are not applicable to multilayer learning because in the *IESVD* and *UESVD* algorithms, the sub-matrices are selected based on less number of items and users are not necessarily included in the larger sub-matrices, which consist of more items and users.

Algorithm 3.6 The Proposed *User-wise ESVD* (*UESVD*)

Input: Rating matrix R $\in \mathbb{R}^{m \times n}$, where $P_{i \in [1,m]} \in \mathbb{R}^{1 \times n}$ is the row vector, κ is a set of elements in the rating matrix that have been signed values; The number of items selected in the sub-matrix based on popularity N and the number of users selected in the sub-matrix based on activity N';

Output: RMSE of the test set;

Step 1: Sort users based on the number of ratings they rates (activity) in descending order $i(1), i(2), ..., i(n)$;

Step 2: Create a sub-matrix M_1 by selecting the top N' users (rows) of R based on the popularity. Therefore $M_1 = [P_{i(1)}, P_{i(2)}, ..., P_{i(N')}](N' < n)$ where $Q_{j \in [1,n]} \in \mathbb{R}^{m \times 1}$ is the column vector of M_1;

Step 3: Sort items based on popularity of the sub-matrix M_1 in descending order $i(1), i(2), ..., i(n)$;

Step 4: Create a sub-matrix M_2 by selecting the top N items (columns) of M_1 based on the popularity. Therefore $M_2 = [Q_{j(1)}, Q_{j(2)}, ..., Q_{j(N)}](N < m)$;

Step 5: Apply basic matrix factorization (*Regularized SVD*) on matrix M_2 to obtain feature matrices U and V according to Equation (3.1); Then predict every missing value in sub-matrix M_2 to acquire a non-null matrix M_2' according to Equation (3.2). Then a series of ratings L is obtained, such that $L = \{r_{i_{k(1)}, j_{t(1)}}, ..., r_{i_{k(n)}, j_{t(n')}}\}$ where $r_{i_k, j_t} \in (M_2 \bigcap \neg \kappa)$;

Step 6: Fill ratings in the original matrix R with every predicted value by **Step 5** to acquire a new rating matrix R$'$. That means the extra ratings are added into the set of existing ratings. $\kappa = \{\kappa, L\}$;

Step 7: Apply basic matrix factorization (*Regularized SVD*) on matrix R$'$ to obtain feature matrices *U'* and *V'* according to Equation (3.1). Then predict the target ratings (test set) according to Equation (3.2) and calculate *RMSE* according to Equation (1.2);

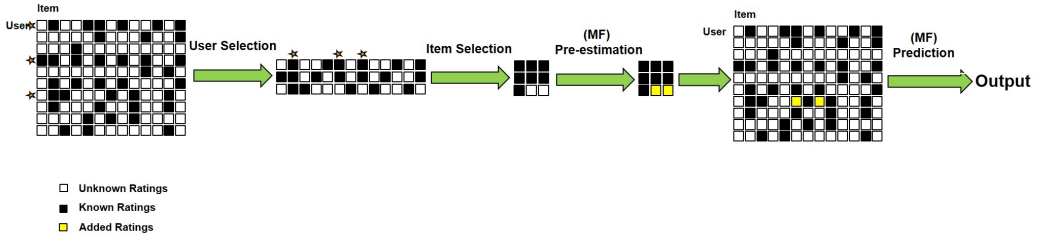


Figure 3.10: Procedures of *User-wise ESVD*

Table 3.5: Experimental datasets

Dataset	Size	Number of ratings	Density
MI	6040×263	59005	3.72%
MU	401×3952	70923	4.46%
NI	6800×500	105444	3.10%
NU	955×3561	110818	3.26%

3.5.3 Experimental Results

To emphasize the benefits of the proposed *IESVD* and *UESVD* approaches, the following two subsets are extracted from *Movielens 1M* to make the size similar to the *Movielens 100K* dataset in the experiments:

1. *MI (*6040×263*)*: This dataset contains ratings of 263 movies which are randomly selected from 3,952 movies provided by 6,040 users.

2. *MU (*401×3952*)*: This dataset contains ratings of 3,952 movies provided by 401 users which are randomly selected from 6,040 users.

Likewise, the following two subsets are also extracted from the original *Netflix* dataset to make the size equal to the *Netflix* subset for comparative purpose.

1. *NI (*6800×500*)*: This dataset contains ratings of randomly selected 500 movies provided by 6,800 users.

2. *NU (*955×3561*)*: This dataset contains ratings of randomly selected 3,561 movies provided by 955 users.

Experiments of the proposed *IESVD*, *UESVD* approaches are conducted on the *Movielens 1M* subsets *MI*, *MU* and *Netflix* subsets *NI*, *NU* where the details are shown in Table 3.5.

Table 3.6 to Table 3.9 show some experimental details of proposed methods on the datasets including the number of selected items and users (N=N'=10% for the *Movielens 1M* subsets *MI*, *MU* and $N = N' = 5\%$ for the *Netflix* subsets *NI*,

Table 3.6: Comparison of the proposed methods on *MI* (6040×263)

N=10%	Block Density	Extra Ratings	*RMSE* (Best)
RSVD	null	null	1.0432
ESVD	53.80%	7255	1.0286
IESVD	56.58%	6818	1.0235
UESVD	54.64%	7123	1.0246

Table 3.7: Comparison of the proposed methods on *MU* (401×3952)

N=10%	Block Density	Extra Ratings	*RMSE* (Best)
RSVD	null	null	0.9898
ESVD	57.52%	6712	0.9791
IESVD	58.52%	6554	0.9749
UESVD	58.80%	6509	0.9802

NU), the density of sub-matrix, the number of added ratings and the results of different algorithms on corresponding datasets.

Specifically, different sub-matrices are first created by following different s-trategies. It can be observed that for the datasets of which the number of users is far greater than the number of items (for datasets *MI* and *NI*), *IESVD* could obtain denser sub-matrices. While for datasets which contain more items than users (for datasets *MU* and *NU*), the density of sub-matrices based on *UESVD* are greater. However, the number of extra ratings predicted from denser sub-matrix is less than the one that are predicted from sparser sub-matrix. Therefore, it is inappropriate to compare the results of different algorithms based on the certain number of items and users N. As the result, the best performance (with least *RMSE*) of proposed algorithms are directly listed based on best choices of N (setting $N = N'$).

Table 3.8: Comparison of the proposed methods on *NI* (6800×500)

N=5%	Block Density	Extra Ratings	*RMSE* (Best)
RSVD	null	null	0.9620
ESVD	59.31%	3459	0.9567
IESVD	66.36%	2859	0.9552
UESVD	61.08%	3308	0.9560

Table 3.9: Comparison of the proposed methods on NU (955×3561)

N=5%	Block Density	Extra Ratings	$RMSE$ (Best)
$RSVD$	null	null	0.9439
$ESVD$	62.54%	2038	0.9400
$IESVD$	68.18%	1717	0.9392
$UESVD$	68.79%	1684	0.9376

Figure 3.11 to Figure 3.14 show the resulting performance ($RMSE$) of the proposed methods based on how many items and users (setting $N = N'$) are selected in the sub-matrix on MI, MU, NI, NU datasets, respectively. As it can be seen from figures that all the algorithms start from zero point where no extra ratings are added into the original matrix, which can be seen as the special case of $RSVD$ for comparison. As the number of items and users selected in the sub-matrix N goes up, the performance fluctuates. When N is getting large, excessive ratings distort the model and deteriorate the performance. Therefore the best choices of N that lead to the least $RMSE$ are compared. It can be observed that when dealing with the datasets MI and NI where the number of user is far greater than the number of items, $IESVD$ yields denser sub-matrix than the $UESVD$ method. When the datasets contain more items than users (MU, NU), $UESVD$ performs better than $IESVD$.

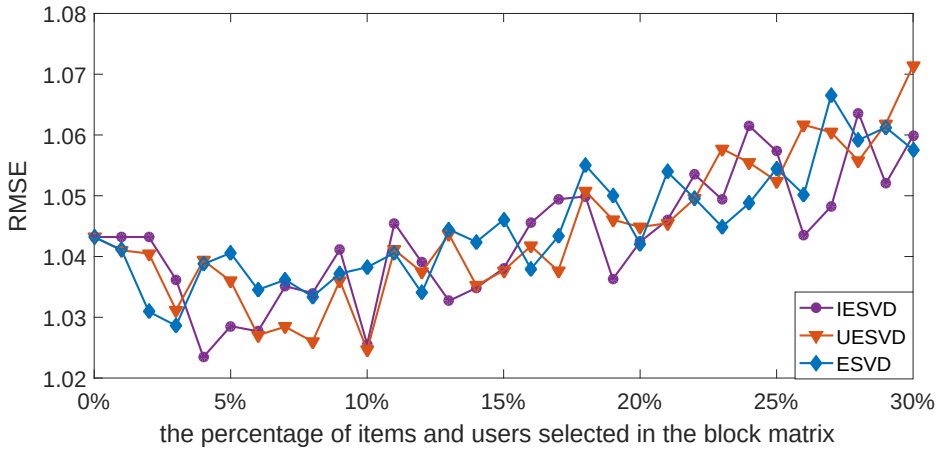


Figure 3.11: *RMSE* of the proposed methods on *MI* (6040×263)

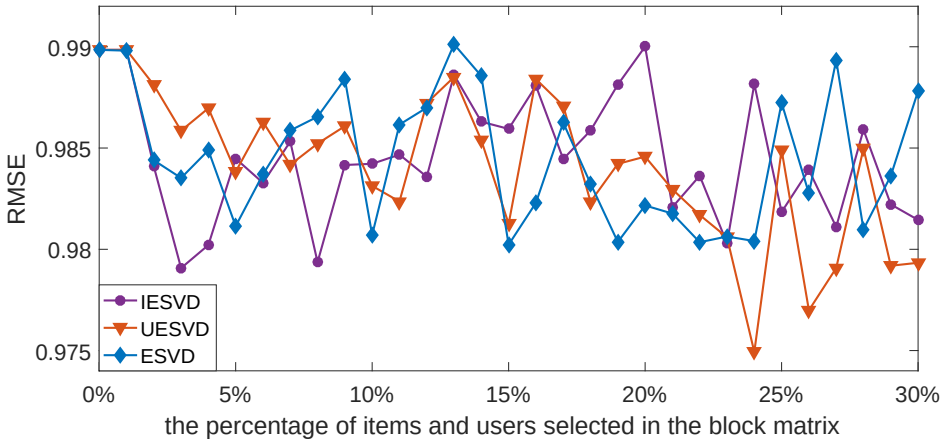


Figure 3.12: *RMSE* of the proposed methods on *MU* (401×3952)

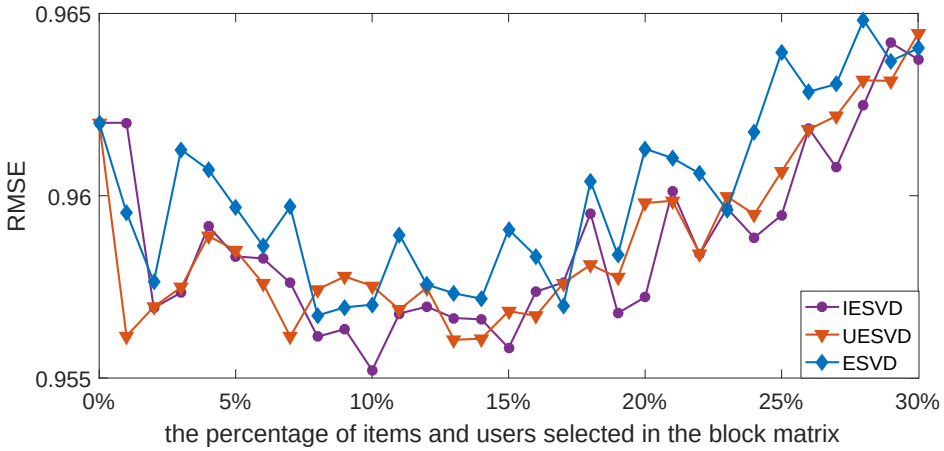


Figure 3.13: *RMSE* of the proposed methods on *NI* (6800×500)

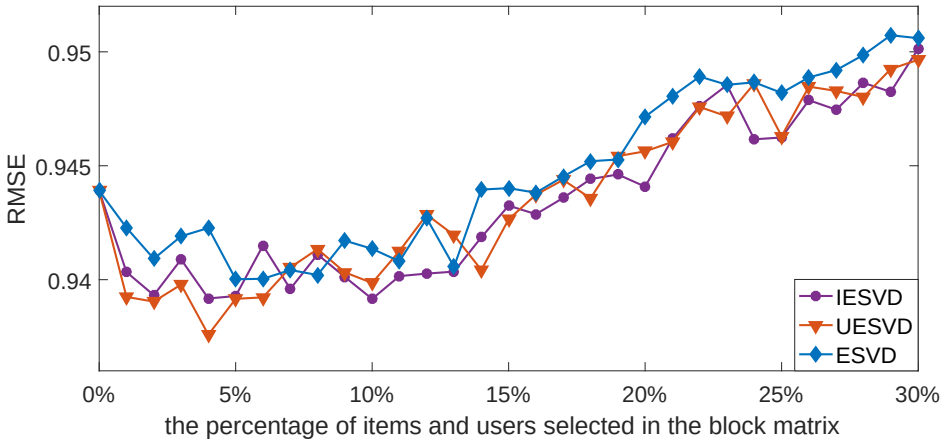


Figure 3.14: *RMSE* of the proposed methods on *NU* (955×3561)

3.6 Summary

The lack of information is an acute challenge in most recommender systems. In this chapter, a series of methods are proposed which apply the traditional matrix factorization method with ratings completion that best approximates a given matrix with missing values.

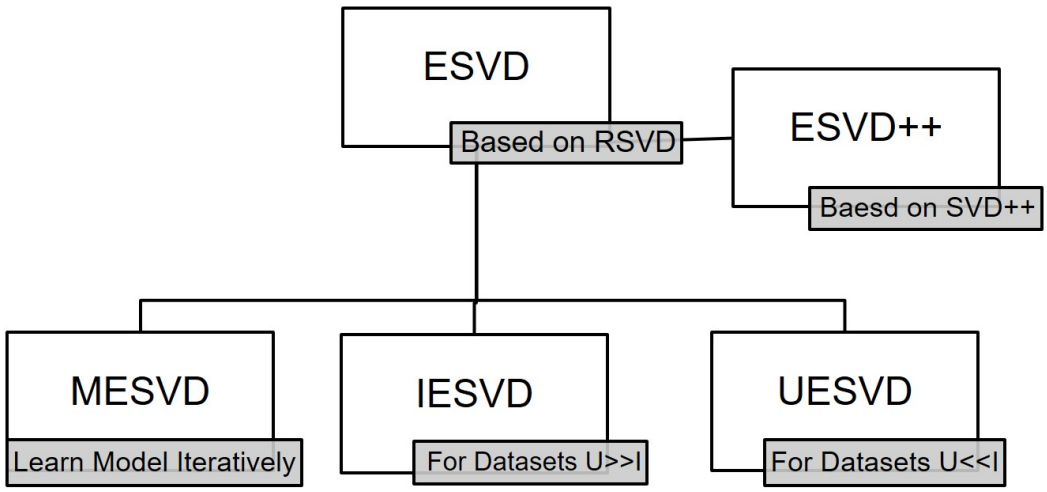


Figure 3.15: The proposed ESVD and its variants

Specifically, the general *EVSD* model is firstly proposed by combining the proposed item-oriented approach and user-oriented approach that inspired by active learning. The corresponding experimental results show its benefits in prediction accuracy. Then this general framework can be incorporated with different *SVD-based* algorithms such as *SVD++* by proposing the *ESVD++* method. The proposed *EVSD* model is further explored by presenting the *MESVD* approach, which learns the model iteratively. This *MESVD* approach achieves better performance than *ESVD* but in sacrifice of training time. In addition, two variants of *ESVD* model are proposed: *IESVD* and *UESVD*. Although the *IESVD* and *UESVD* approaches can

not be learnt through multilayer learning strategy like *MESVD*, their performance are better than *ESVD* for handling the imbalanced datasets that contains more users than items or more items than users, respectively.

Instead of viewing active learning from the individual user's point of view, the proposed methods deal with the problem from the system's perspective. Also, they tackle the problem of active learning of which the query process is costly and unrealistic. Although the proposed methods cannot deal with the cold start problem where the database keeps growing as new users or items continue to be added, it does reduce the computational cost greatly since all the ratings are added simultaneously (*ESVD*, *IESVD* and *UESVD*) or iteratively by a predefined number of times (*MESVD*).

Chapter 4

A Generalized Framework of System-Driven Active Learning in Collaborative Filtering Recommender Systems

4.1 Problem Statement and Motivation

Collaborative filtering recommender systems predict other items that users might like based on the knowledge of preferences (usually expressed in ratings) of users for some items. The performance of collaborative filtering recommender systems, given a certain amount of ratings, depends on prediction algorithms. There are two primary prediction algorithms to deal with collaborative filtering: neighborhood approaches (memory-based algorithms) and latent factor models (model-based algorithms). Neighborhood methods [50] concentrate on the relationship between items or users, so they are good at detecting localized relationships. By transforming both items and users to the same latent space, latent factor models try to explain ratings

by items and users, aiming at making them directly comparable. Generally, matrix factorization, as one of the most successful realizations of latent factor models, can produce better accuracy than classic nearest neighbor methods when dealing with product recommendations because of the incorporation of additional information such as implicit feedback and temporal effects [36].

Apart from prediction algorithms, the performance of collaborative filtering recommender systems also rely on the knowledge (e.g. ratings) that users provided regarding items. Especially during sign-up process, the systems usually find difficulties in making recommendations for users who were recently introduced into the systems. To overcome this issue (cold start problem [160]), some systems would first ask users to rate a given set of items for better recommendations. However, obtaining information from users is costly since users are often unwilling to rate a large amount of items. Therefore active learning for collaborative filtering is proposed to acquire high quality data that help most in representing the interests of the users. To achieve this purpose, the system requests the user to rate specific items based on certain strategies or criteria. The ultimate goal is to get the maximized error reduction with the least queries for the target user. In summary, traditional active learning for collaborative filtering is a set of techniques that select a number of items to rate, so as to improve the rating prediction for the user. On the other hand, Carenini et al. [21] proposed an item-focused method that elicits ratings by choosing some special users to rate a specific item in order to improve the rating prediction for this item.

However, traditional active learning methods [21] [128] [129] [131] only evaluate each user or item independently and only consider the benefits of the elicitations to new users or items, but pay less attention to the effects of the system. In addition, ratings were added one by one per request [129] or user's by user's per request [131], which incurs high computational cost. In this chapter, a novel generalized framework is proposed for applying active learning in recommender systems. Specifically,

the elicitations of the past users are focused instead of the new users, and a more general scenario, where users repeatedly come back to the system instead of only during the sign-up process, is considered. Furthermore, in the proposed framework, the ratings are elicited simultaneously based on the criteria with regard to both items and users, for the purpose of improving the performance of the whole system. In addition, a variety of active learning strategies are tested on the proposed framework based on the matrix factorization method and finally has shown that this framework can be expanded to the conventional active learning with specific settings.

4.2 Traditional Active Learning in Collaborative Filtering

Most recommender systems suffer from the cold start problem: when a new user comes in, the recommender system has little knowledge about the user. Therefore it is difficult to provide proper suggestions given the circumstance. To tackle this issue, active learning was proposed by asking users to rate a set of preselected items during the enrollment stage [161].

In the early work of [115], Merialdo et al. first proposed to use *Entropy* and *Variance* as active learning strategies for rating elicitation, and showed that through this smart selection the recommender system achieves better performance for a certain amount of ratings required from the user, or reduce the amount of elicitations to achieve the given performance when compared with random selection based on the neighbourhood algorithms [49]. Rashild et al. [20] extended this work by introducing and comparing six strategies: the *Entropy* strategy selects items with the largest entropy; the *Random* strategy, which selects items to present randomly with uniform probability over all the items; the *Popularity* strategy where items with the largest number of ratings are preferred; the *Popularity*Entropy* strategy, which considers

both popularity and entropy; the *Log(Popularity*Entropy)* strategy, which takes the log of the ratings that linearized popularity, making it a better match for entropy; and the *Item-Item Personalized* strategy, which presents movies using any strategies until the user has given at least one rating, then selects items that the user is likely to have seen by computing similarity between items. All the strategies were also tested based on the neighbourhood models [49] and the *Log(Popularity*Entropy)* strategy was found to be the best for reducing the *Mean Absolute Error* (*MAE*) of predictions regarding the new users. Later in [117], the same authors further explored their work of [20] by proposing three strategies. The *Entropy0* strategy is an extension of the *Entropy* strategy, where the missing values are considered to be 0 as a single category. The *Harmonic mean of Entropy and Logarithm of rating Frequency (HELF)* strategy is for finding items that are familiar with others and with high variability. The *Information Gain through Clustered Neighbours (IGCN)* strategy was proposed based on decision trees where each node is labelled by a particular item. Users are clustered into groups with similar profiles and items with the largest information gain by considering all users or neighbors in the same cluster are elicited in different stages. They focused on the elicitation strategies for the completely new users, and the performance was evaluated only on these new users by neighbourhood algorithms [49]. In contrast, this work concentrates on the rating elicitation for users who pre-entered into the systems, and evaluate the performance of the whole system by the matrix factorization method.

Carenini et al. [21] pointed it out that users can give elicitations whenever she or he is motivated, therefore they presented the *Conversational and Collaborative Interaction* model where ratings could be elicited from both new users and existing users. The authors also proposed the item-focused approach that elicits ratings to improve the rating prediction for a specific item. However, they only utilized the popularity-based and entropy-based strategies for items or users seperately and the performance was evaluated on specific users or items who has elicitations, respec-

tively. In contrast, this work tests a variety of strategies simultaneously for both items and users in the system-wide perspective.

Later in [120], Golbandi et al. introduced the *Coverage* strategy. It selects items with the largest coverage, which is defined as the total number of users who co-rated both the selected item and any other items. In addition, *GreedyExtend* strategy was proposed, where the items that minimize the *RMSE* of the predictions on the training set are selected. Furthermore, they also presented the *Sqrt(Popularity)*Variance* strategy that finds items with diverse and a large number of ratings. And finally the *Voting* strategy, which considers the overall effect of previous methods, was also proposed by the same authors [120]. In their works, ratings were only elicited one by one for each user or user by user. Again they only tested the improvements of prediction accuracy for particular users who have elicited ratings. In my experiments, ratings are elicited simultaneously, and the performance is evaluated based on the whole systems.

4.3 The System-Driven Active Learning in Collaborative Filtering

Most early works on active learning in collaborative filtering implemented different elicitation strategies based on the classic machine learning methods such as neighbourhood methods [115] [117] or Bayesian learning based aspect models [128] [129]. Recently matrix factorization methods [36] have been widely used and achieved promising prediction accuracy in recommender systems. Matrix factorization methods have also been explored in active learning scenarios such as [130] and [162]. However, these works still concentrated on the elicitation strategies for new users only.

4.3.1 The Proposed Generalized Framework

In more recent work of [131], Elahi et al. proposed that the rating elicitations of users not only improve the prediction of the target user but also help the system to give suggestions for other users. They evaluated active learning strategies in the system-wide perspective to test how different elicitation strategies for users affect the performance of the whole system. In their work they simply utilized the matrix factorization method as the prediction algorithm to show that elicited rating has effects across the system based on their experimental results, but fails to build connections between them.

Actually the elicitation of the system-wide effects is not applicable to all the scenarios. For example, in classic item-based neighbourhood method [49], an elicited rating of an item can only affect the prediction of its neighbours. As rating matrices are often extremely sparse, most items have no correlation to the elicited items, therefore the elicitation cannot influence the recommendations of the users who have not rated elicited items.

The rationale of system-wide effectiveness is that matrix factorization methods decompose the rating matrix in the products of two side matrices which consist of feature vectors corresponding to items and users. Therefore each user's rating is composed of the sum of preferences about the various latent factors of that item. Since the parameters (latent factors) of the model are learnt by fitting a limited number of existing ratings (details can be found in Section 3.2.1), each elicitation of ratings would inevitably affect the parameters learning in the matrix factorization models, and further influence the predictions of all the users in the system.

In previous works, active learning strategies were only implemented as criteria for selecting specific items for each user. In other words, the elicitation has no limitations for users. In fact, each user may act differently when asked to provide ratings. There are two metrics that are usually taken into consideration in active learning scenarios. The first one is the number of elicited ratings, which depends

on whether the user will give ratings to the queried items or not. For example in movie recommendation scenarios, though the active users who are enthusiastic about movies may watch far more than the ones who are not into movies, there still exist some movies the users have watched but not yet rated. Therefore, it is easier to accept that active users have high possibility to give ratings to the movies when asked to, but little chance for the users who had no interest in providing ratings (with a small number of ratings in the data set). The second consideration is the quality of the elicitations. For example, the elicitations of the users who used to give nearly even ratings or extremely random ratings for items have little or even negative effects on helping rating predictions. Therefore, querying the critical users who take it seriously for rating elicitation would be preferred.

Algorithm 4.1 The Proposed System-Driven Active Learning Framework

Input: A set of elements κ in the rating matrix that have been assigned values; a set of ratings ϕ that are known by the users; a test set τ which consists of a number of ratings that are supposed to be predicted by the system; Predefined itertation time K;

Output: Evaluation (often measured by *RMSE*, *MAE*, etc.) of the test set τ ;

In each iteration:

Step 1: Select a set of ratings $\chi_1 \in \tau$ based on a predefined item selection criterion (active learning strategy);

Step 2: Select a set of ratings $\chi_2 \in \tau$ based on a predefined user selection criterion (active learning strategy);

Step 3: Only the ratings that are both selected from Step 1 and Step 2 are considered as elicitations, in this case $\chi_3 = \chi_1 \cap \chi_2$;

Step 4: Add the selected rating (or ratings) from Step 3 into the training set, therefore $\kappa = \{\kappa, \chi_3\}$;

Step 6: Remove the selected ratings χ_3 from the learning set ϕ;

Step 7: Train the prediction model (matrix factorization in this case) based on the updated training set κ;

Step 8: Evaluate the predictions in test set τ based on the trained prediction model.

Step 9: Repeat Step 1 to Step 8 for K times;

A system-driven active learning is proposed in *Algorithm* 4.1 which incorporates a conventional user-focused active learning with the items-focused active learning, trying to improve the performance of the whole system based on the ma-

trix factorization method. Traditional active learning elicit ratings based on different item selection strategies for each user. In contrast, the ratings in the proposed framework are not only elicited based on the traditional item selection strategy, but also need to fulfill the user selection strategy. Therefore, in each iteration only the intersections of the elicited ratings that are both selected based on the item selection criterion and user selection criterion are elicited from the learning set to the training set (Step 1 to Step 6). In this case, the system will only query the qualified users for rating elicitations on specific items. Since the ratings of users are used as a source of information for picking candidates, the elicitation process is only for the users who have entered the system. It is based on the assumption that past users would repeatedly come back to the system for receiving recommendations, and give elicitations when the system queries. Also, elicitations must take into consideration that users are willing or not to answer such queries. For example, if an user has not watched queried movies, he or she is not able to provide the rating for this movie. Therefore, only the ratings known by the user (in the learning set) are elicited. In each iteration new ratings are added from the learning set in the training set based on the different elicitation strategies (which will be introduced in next section). Instead of evaluating only the new users in the traditional active learning, the benefits to the system are considered by evaluating all the users in the systems (test set).

Most users are interested to see the response (e.g. changes) of recommendations immediately in the process of eliciting, which would stimulate them to give more ratings in turn. For this reason, many traditional active learning algorithms are implemented by sequential learning where all the ratings are elicited incrementally based on a certain elicitation strategy. Therefore, the model is re-trained and the elicitation strategy is updated whenever the rating is added. As a result, the system would generate more appropriate recommendations for user. However, sequential learning is not practical since retraining the model for each rating is very time consuming. Therefore, batch learning is often used by readjusting the model

after users have elicited several items.

In this work, traditional user-focused active learning is incorporated with item-focused active learning. Therefore, ratings are selected as the intersections of two rating sets with the user selection criteria and the item selection criteria, respectively. As a result, a batch of ratings, in most cases, will be elicited simultaneously. The second consideration is that, the benefits of elicitations is evaluated to all the user (system-wide), while a single elicitation only produces trivial effect on the performance of the whole system. Therefore, the experiments are implemented by batch learning.

4.3.2 Active Learning Strategies

An active learning strategy in collaborative filtering is the procedure for selecting which items to present to the user for rating elicitation. Several traditional active learning strategies [131] [117] [20] [120] [19] have been proposed and evaluated in the collaborative filtering recommender systems. In this work, a novel approach is proposed which incorporates the user selection criteria into the tradition active learning which only focuses on the way to selecting items. Based on this framework, strategies which are applicable to both items and users that will contribute to the improvement of the system performance need to be identified. These strategies can be divided into two categories: single-heuristic or combined-heuristic, depending on whether the strategy takes into account a single criterion or combines a number of criteria.

4.3.2.1 Single-Heuristic strategies

Single-heuristic strategies are based on one heuristic by utilizing the unique selection rule for both items and users.

- *Random* [20]: selects items or users to present randomly with uniform probability over all the items or users, which can be regarded as the baseline strategy

for comparison.

- *Frequency* [20]: items or users with the largest number of ratings are preferred. The more ratings an item has been rated, the more popular this item is. Therefore, it is more likely that a user is able to give ratings to popular items. As for users, it is also easy to accept that active users who used to be interested in rating items are more likely to give more ratings when the system queries. However, frequency-based methods elicit ratings regarding the popular items and the active users will lead to corresponding items and users more popular and active in the system, respectively.

- *Variance* [20]: selects items or users with the largest variance for eliciting. The variance of an item is calculated as:

$$Variance(i) = \frac{1}{|U_i|} \sum_{u \in U_i} p(r_{ui} - \bar{r}_i)^2 \tag{4.1}$$

where $\bar{r}_i$ is the mean ratings of item i, and U_i is the set of users who rated this item. p is the probability mass function.

Variance is maximized when ratings deviate the most from mean ratings. This strategy is based on the assumption that the system is supposed to be uncertain about the items with diverse ratings which represent the preferences of users. Therefore the items with the largest variance are preferred for reducing the certainty of the system. The users with the largest variance are supposed to give their opinions discriminatively, who are also preferred.

- *Entropy* [20]: selects items or users with the largest entropy which are considered to be informative. Entropy is computed by using the relative frequency of each of the five possible ratings (1-5).

$$Entropy(i) = -\sum_{k=1}^{5} p(r_i = k) log(p(r_i = k)) \tag{4.2}$$

where $p(r_i = k)$ is the probability that a user rate the item i as k.

It measures the dispersion of the ratings a user has rated or the item has been rated, and is maximized when all the ratings are equally likely. However the *Entropy* strategy has the tendency to choose unpopular items or inactive users since items or users with only few ratings may result in large entropy, especially in extremely sparse rating matrices.

- *Entropy0* [117]: tackles the problem of the *Entropy* strategy that tends to select unpopular items or inactive users by assigning all the missing ratings to 0. Therefore, the unpopular items or the inactive users with few ratings will result in small entropy, which are not taken into consideration by this strategy.

$$Entropy0(i) = -\sum_{k=0}^{5} p(r_i = k) log(p(r_i = k)) \tag{4.3}$$

4.3.2.2 Combined-Heuristic strategies

Combined-heuristic strategies implement multiple selection rules for items and users by aggregating and combining a number of single-heuristic strategies, in order to achieve a range of objectives.

- *Log(Frequency)*Entropy*: considers both frequency and entropy, trying to collect a large number of ratings with rich informativeness for items or users. This strategy takes the log of the ratings that linearizes frequency, making it a better match for entropy.

- *Sqrt(Frequency)*Variance*: amplifies variance by multiplying it with the square root of the item or user frequency, trying to find items or users with diverse and a large number of ratings.

- *Coverage* [120]: selects the items or the users with the largest coverage.

Suppose $\mathrm{R} \in \mathbb{R}^{m \times n}$ is the rating matrix of m users and n items. The coverage of an item i is calculated as:

$$Coverage(i) = \sum_{j=1}^{n} I_{ij} \tag{4.4}$$

where I_{ij} is the number of users who have rated both item i and item j.

This strategy captures the items highly co-rated by users or the users that have the most co-rated items based on the assumption that eliciting their ratings may improve the prediction accuracy for the other items or users.

- *HELF* [117]: stands for *Harmonic mean of Entropy and Logarithm of rating Frequency*, which is defined as:

$$HELF(i) = \frac{2 \times LF(i) \times H(i)}{LF(i) + H(i)} \tag{4.5}$$

where *LF(i)* is the normalized logarithm of the rating frequency and *H(i)* is the normalized entropy of the item or user i.

This strategy takes both entropy and frequency into consideration by using the harmonic mean (harmonic mean is high when both factors are high), trying to select informative items or users that also have a large number of ratings.

4.4 Evaluations of the Proposed Framework

4.4.1 Datasets and Experimental Setup

Experiments are conducted on the classic recommender system datasets: *Movielens 100K* and the subset of the *Netflix*. Some experiments with the larger version are also performed and obtained similar results. However, it requires much longer time to perform the experiments since the model is trained and tested in each iteration as more ratings are being elicited. Therefore, the smaller datasets *Movielens 100K* and

the subset of the original *Netflix* are focused to be able to run more experiments, in order to explore how rating elicitations affect the performance of the whole system.

For both datasets all the known ratings are partitioned randomly into three sets:

- Training set: contains 20% of the ratings, which are considered as known by the system. The ratings in this dataset are used for training the matrix factorization model in each iteration in the active learning process.

- Learning set: contains 60% of the ratings, which are regarded as known by the users but not known by the system. Therefore the ratings in this dataset are elicited incrementally to the training set if the system queries.

- Test set: contains 20% of the ratings that are used to evaluate the elicitation strategies.

In the experiments the number of queried items and users are set to be from 0% to 100% with 1% increase (simply setting items equal users in percentage) in each iteration based on different strategies. Therefore the number of iterations is 101 from the stage of training the model with no elicitation to the stage with all the elicitations. Then all the ratings in the learning set are elicited incrementally to the training set, which is utilized for building the matrix factorization model. The number of latent factors (rank) k are set to be 10 for training the matrix factorization model [36]. Although increasing it does raise the performance, the computational cost is proportional to the latent factors. At last, the performance of system is evaluated by comparing the difference between the predictions from the model (matrix factorization) and the ground truth in test set, usually measured by their *RMSE*.

These settings are based on the assumption that the recommender system was first built by a small dataset (20%). Afterwards it keeps obtaining ratings by querying different users about different items based on corresponding strategies. By

iteratively acquiring knowledge from elicitations (from 20% to 80%), the system can generate more precise recommendations.

4.4.2 Performance Analyses

In this section the results of the experiments are presented based on three aspects: system *RMSE* evolution, elicited ratings evolution and the quality of elicited ratings.

4.4.2.1 *RMSE* - Iteration

In the proposed active learning framework, ratings are elicited by querying a set of users for ratings about certain items iteratively through batch learning. We first present how the system *RMSE* is changing with the training dataset keeps acquiring more and more elicited ratings in each iteration according to the different strategies based on the proposed framework.

The model is trained by starting from iteration 0 where no rating is elicited in the system, and finishing at iteration 100 with all the ratings from learning set added into the system by different strategies, which models users come back to the system at each iteration. Therefore the same results are obtained from these two specific points for all the strategies.

From Figure 4.1 and Figure 4.2 it can be observed that the *Random* strategy decreases *RMSE* gradually for both datasets. For the *Movielens 100K* dataset, the performance of all the strategies fluctuates significantly in the early stages (as shown in Figure 4.1). From iteration 16 to 66, the *Log(Frequency)*Entropy* and *Sqrt(Frequency)*Variance* strategies obtain the best performances, while *Frequency*, *Entropy0* and *Coverage* generate poor outcomes. After that the best strategies are overtaken by the *Coverage* and *Entropy0* strategies.

For the *Netflix* dataset, the performance of most strategies remain relatively steady at the initial stages (as shown in Figure 4.2). This happens because the *Netflix* dataset is much sparser than the *Movielens 100K* dataset. In the beginning,

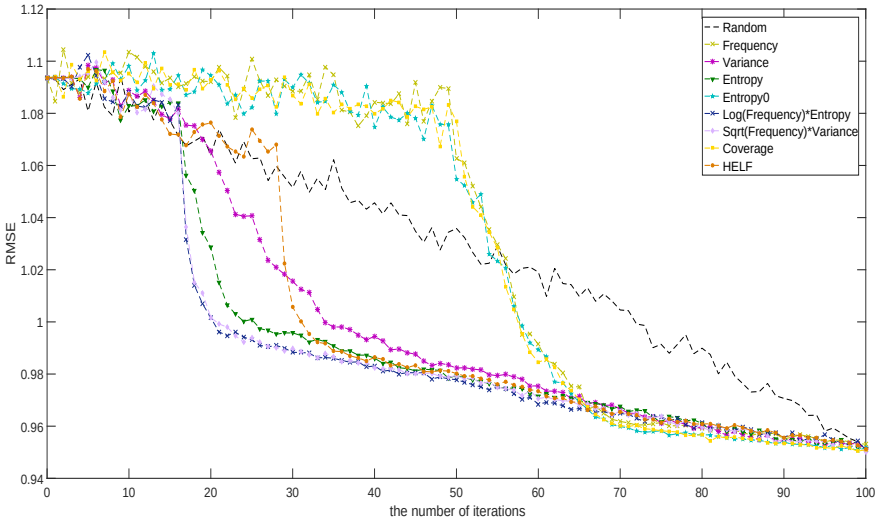


Figure 4.1: System *RMSE* evolution based on the learning process on *Movielens 100K*

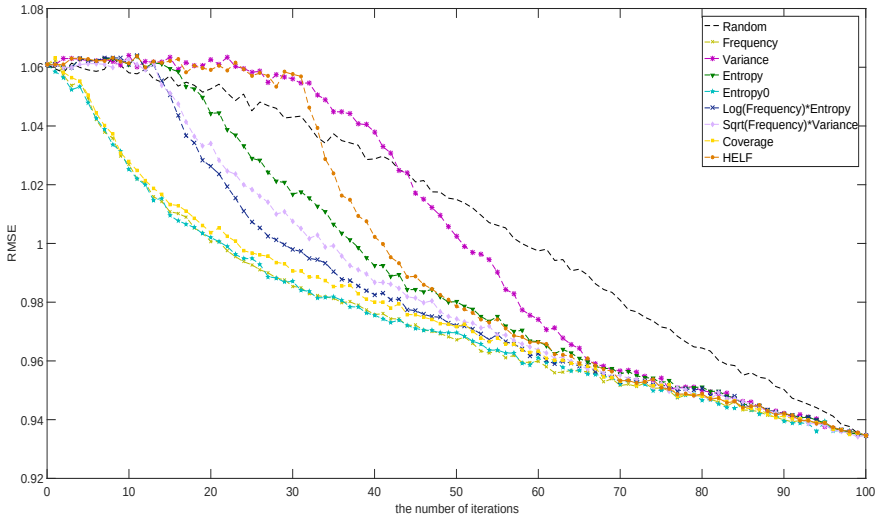


Figure 4.2: System *RMSE* evolution based on the learning process on *Netflix*

since queried items and users are few, only a limited number of ratings are elicited, which produces little effect on the performance of the whole system. The performance of the *Frequency*, *Coverage* and *Entropy0* strategies drop rapidly. While the *RMSE* of other strategies remain steady in the first 15 iterations, then start to decrease continuously. All the strategies achieve better performance than randomized selection strategy after iteration 45, then generate similar results after iteration 66. *Entropy0* is considered to be the best strategy for the *Netflix* dataset.

4.4.2.2 *Number of Elicited Ratings* - Iteration

The number of elicited ratings varies depending on the type of the elicitation strategy. Through proper strategies more ratings can be obtained by estimating what items some users have actually experienced and are able to give ratings. A larger number of elicited ratings mean that the target users are willing to answer the queried items while a small number of elicited ratings may lead to the frustrating feelings of the users who are not able to rate. Therefore, the number of ratings elicited in each iteration for different strategies are reported based on the proposed framework.

Figure 4.3 and Figure 4.4 show the number of ratings that are elicited from the learning set of the *Movielens 100K* and *Netflix* datasets, respectively. As mentioned before, in the first several iterations only a small number of ratings are queried. As a result, the number of elicited ratings increases slowly for both datasets in the beginning. At the final point all the ratings in the learning sets are elicited.

For the *Movielens* dataset, the best performing strategy are *Frequency*, *Entropy0* and *Coverage* before the first 30 iterations, since these strategies tend to select the most popular items with highly co-rated users and active users with highly co-rated items. Then *Log(Frequency)*Entropy* obtains the largest number of elicited ratings by combining the frequency and informativeness factors. For the *Netflix* dataset, experiments based on *Frequency* and *Entropy0* can acquire more

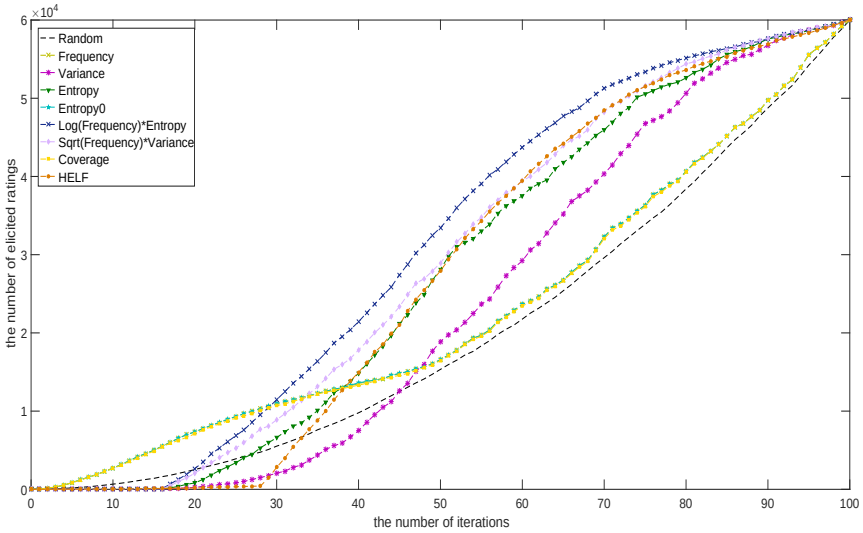


Figure 4.3: Elicited ratings evolution on *Movielens 100K*

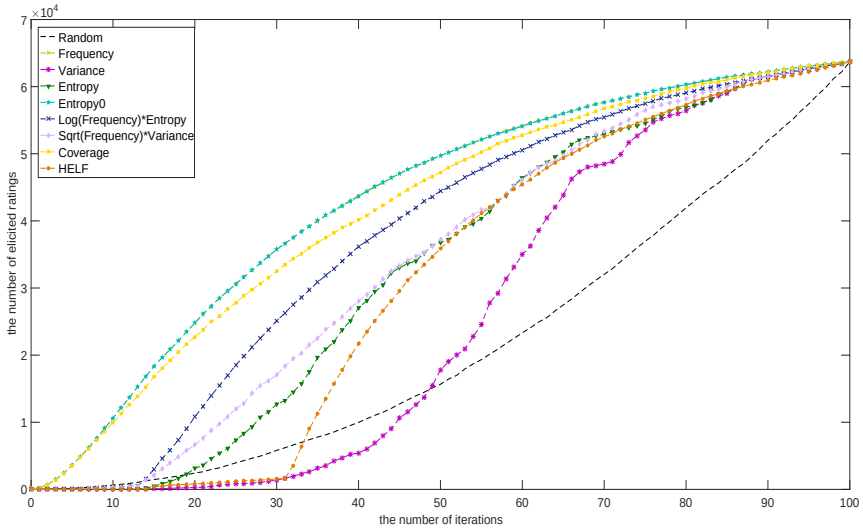


Figure 4.4: Elicited ratings evolution on *Netflix*

ratings than other strategies by filtering unpopular items and inactive users.

4.4.2.3 *RMSE* - Number of Elicited Ratings

Since different number of ratings are elicited based on different strategies in each iteration, it is not appropriate to evaluate each strategy only based on the evolution of experiments. In most active learning works, the quality of the elicited ratings is the priority to be concerned since labeling work is costly. In this work the performance of all the strategies in terms of prediction error (*RMSE*) versus the number of elicited ratings for the *Movielens 100K* and *Netflix* datasets are also reported in Figure 4.5 and Figure 4.6, respectively, in order to find the ratings that minimize the largest *RMSE* through certain elicitation strategies.

For the *Movielens 100K*, the ratings acquired by the *Variance*, *Entropy*, *HELF*, *Log(Frequency)*Entropy* and *Sqrt(Frequency)*Variance* strategies result in lower *RMSE* than the *Random* strategy in most cases. The *Frequency*, *Entropy0* and *Coverage* strategies generate poor performances than the *Random* strategy when the number of elicited ratings is less than 20,000. Beyond the point with more ratings being elicited, the *Entropy0* and *Coverage* strtegies perform best among all the strategies.

For the *Netflix*, all the strategies reduce *RMSE* gradually with ratings added into the training set, thus generating similar results to randomized selection strategy. Most strategies in the early stages still achieve better performance than the *Random* strategy, in which the *Variance* strategy performs best.

The major difference between these two datasets is the sparsity: the *Movielens 100K* dataset contains 6.3% of the possible ratings, and the *Netflix* dataset only contains 3.1% ratings. Since applying strategies has less effects on the sparser dataset, the performances are similar to the *Random* strategy for the *Netflix* dataset.

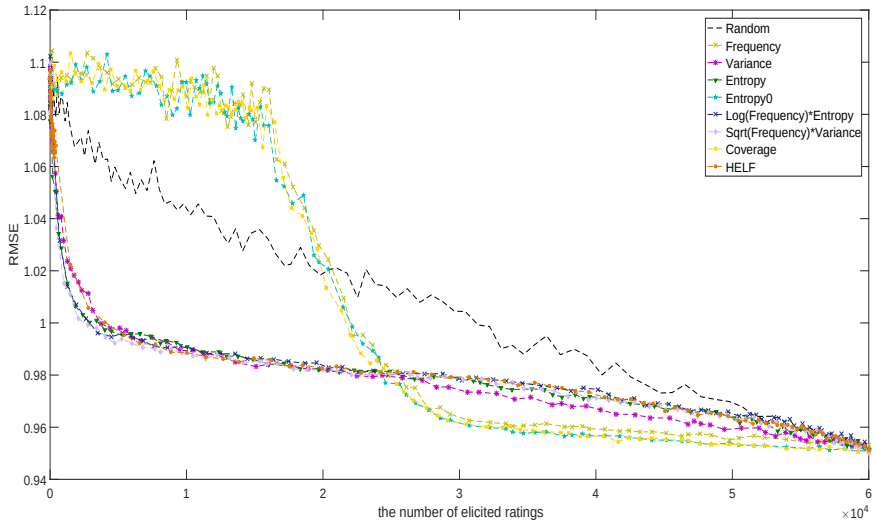


Figure 4.5: System *RMSE* evolution versus the number of elicited ratings on *Movielens 100K*

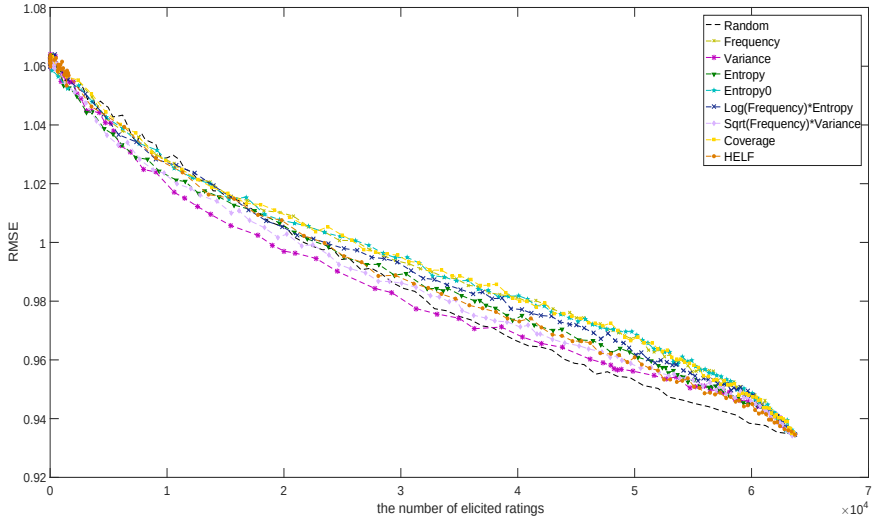


Figure 4.6: System *RMSE* evolution versus the number of elicited ratings on *Netflix*

4.4.3 Comparison with Traditional Active Learning

In the proposed framework, the system queries ratings by incorporating the user selection strategies into the traditional item selection strategies. Hence the ratings are elicited for certain items only by certain users based on different strategies iteratively. Recall that the traditional active learning [117] [124] [131] is a set of techniques that elicited ratings for each user. Therefore it can be regarded as a special case when the user selection criteria are loose to all the users. In this section experiments are also conducted by setting the user strategies as all the users for comparison. Specifically, the number of selected items is from 1% to 100% with a 1% step in each iteration. The number of selected users is 100% in each iteration, meaning that all the users are queried for selected items based on different strategies.

In the experiments of the proposed framework, the ratings are elicited by querying only a set of users about certain items (both from 1% to 100%)s. The number of the elicitations in each iteration depends on the number of mappings of the learning set. Therefore, it is not appropriate to compare the number of elicited ratings or the system performance in each iteration of these two approaches. Since the quality of elicitations (i.e. ratings that lower the largest *RMSE*) is the priority concern in most recommender systems, experiments are conducted by comparing the system performance (*RMSE*) against the number of elicited ratings for both algorithms (combined selection vs item selection).

Figure 4.7 shows the system performance in terms of *RMSE* is relating to the elicited ratings for different strategies (*Frequency*, *Variance*, *Entropy*, *Entropy0*, *Log(Frequency)*Entropy*, *Sqrt(Frequency)*Variance*, *Coverage* and *HELF*) using the *Movielens 100K* dataset. Specifically, the proposed method produces worse performances than the traditional active learning [131] (only apply the item selection strategy) at the initial stages in terms of *Frequency*, *Entropy0* and *Coverage*. This occurs because these strategies tend to select items with a large number of ratings. Incorporating this property into the user selection strategy will accelerate

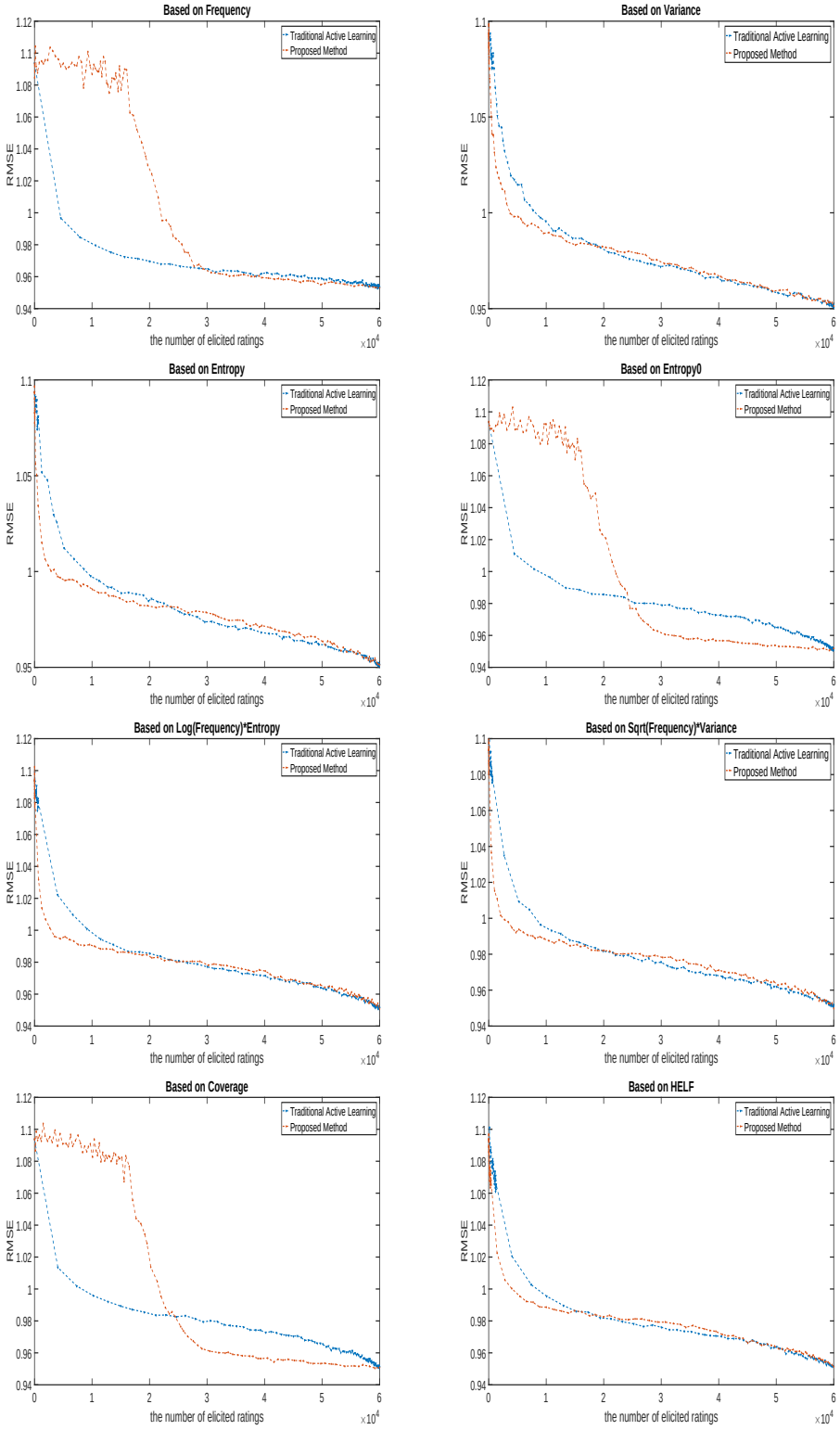


Figure 4.7: System *RMSE* comparison on *Movielens 100K*

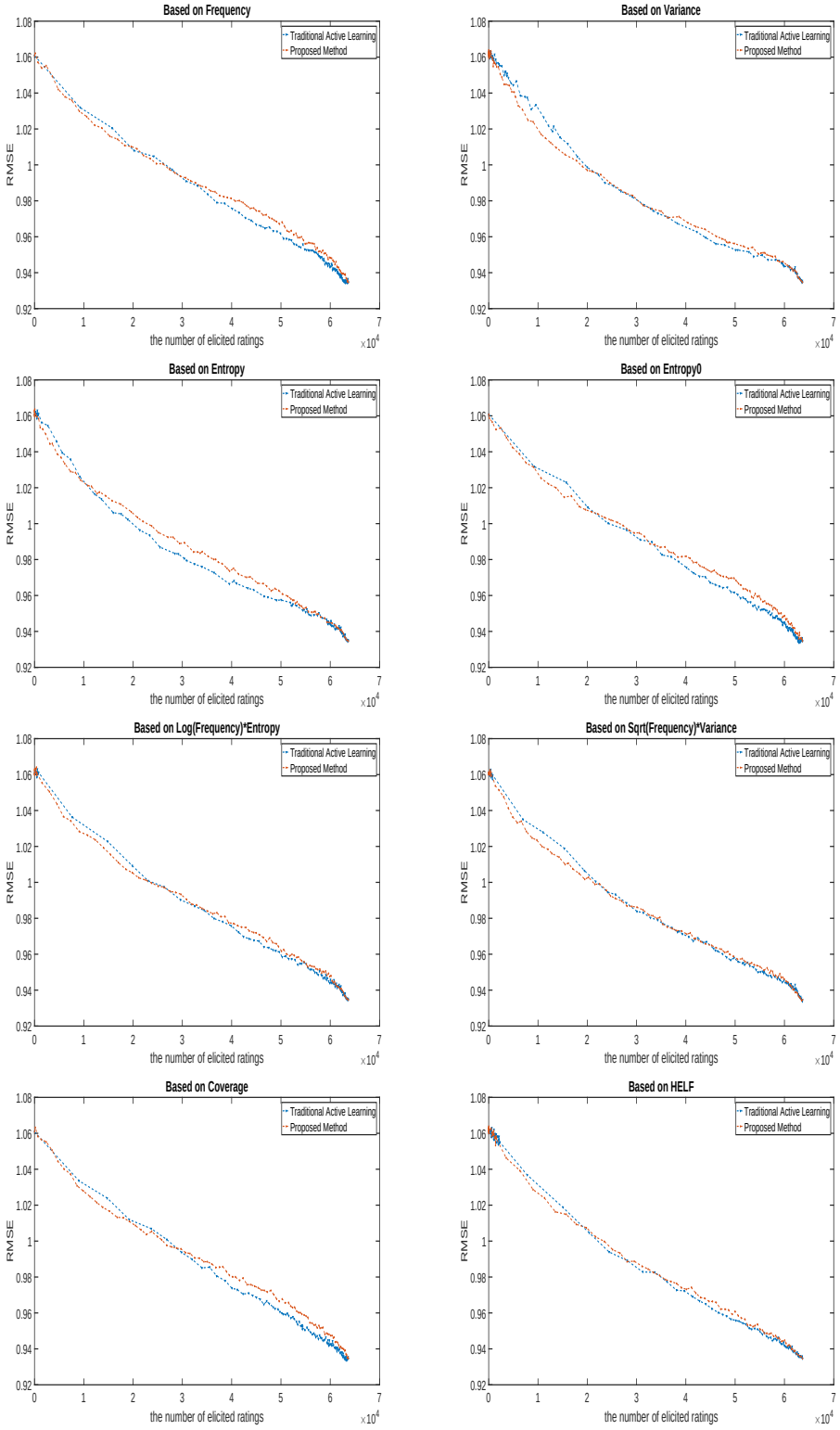


Figure 4.8: System *RMSE* comparison on *Netflix*

the selection bias, which may negatively affect the system performance. For the *Variance*, *Entropy*, *Log(Frequency)*Entropy*, *Sqrt(Frequency)*Variance* and *HELF* strategies, the proposed method outperforms the traditional approach by incorporating more informative and diverse ratings through specific user selection.

The experiments on the *Netflix* dataset are shown in Figure 4.8. Compared with the *Movielens 100K* dataset using the *Frequency*, *Entropy0* and *Coverage* strategies deteriorate the system performance, the *Netflix* dataset reduces the effect of selection bias with extreme sparsity. Although adding user selection criterion has little effects on the *Netflix* dataset since it is much sparser than the *Movielens* dataset, it is apparent that all the strategies based on the proposed framework achieve better performance in the early stages than the ones based on only item selection strategies.

4.5 Summary

Sparsity is a common problem in relating to recommender systems, and the prediction algorithms would fail to give proper suggestions to users without sufficient data. To address this issue, active learning methods are widely used by eliciting ratings from users. In this chapter, a novel generalized active learning framework is proposed which effectively elicits ratings from users for the purpose of improving the performance of the whole system. In addition, it saves computational cost by eliciting multiple ratings simultaneously through batch learning, when compared with traditional active learning algorithms.

The proposed framework is evaluated based on the various strategies in terms of the system performance evolution, the number of acquired ratings and the quality of elicitation. The evaluation has shown that different strategies can improve different aspects of the recommender system in different stages for different datasets.

Finally, the framework has been expanded to conventional active learning

with specific settings (no limitation for users), corresponding experiments including comparisons with the proposed framework are also conducted. The experimental results have shown that the proposed framework could achieve better performance based on certain strategies by taking extra information (users' ratings) into consideration.

However, this method (without specific settings) cannot deal with the cold start problem where the database keeps growing as new users or items continue to be added, since past ratings regarding users and items are used as a source of information.

Chapter 5

Active Learning in Cross-Domain Collaborative Filtering for Sparsity Reduction

5.1 Problem Statement and Motivation

Collaborative filtering is an effective recommender system approach that predicts a user's preferences (ratings) on an item based on the previous preferences of other users. The performance of collaborative filtering suffers from the sparsity problem since each user in the system typically rates very few times and hence the rating matrix is extremely sparse. Even the best algorithms will fail to generate proper recommendations without sufficient knowledge. In practice, borrowing useful knowledge from another rating matrix in a different domain may help producing better recommendations. Fox example, the products in the *Movie* domain and the *Book* domain may have common in genre, therefore it would be useful to make movie recommendations for a user by exploiting his/her preferences on books from the corresponding genre, and vice versa. Therefore, rather than exploiting preferences from each single domain independently, users' preferences knowledge could be transferred

and shared among related domains. This is referred to as cross-domain recommendation [133]. The goal of cross-domain recommendation is to utilize the knowledge derived from the auxiliary domain(s) with sufficient ratings to alleviate the data sparsity in the target domain. A special case of cross-domain recommendation is multi-domain recommendation [24] that utilizes the shared knowledge across multiple domains when all domains suffer from the data sparsity problem, to alleviate the data sparsity in all domains.

Another common way to tackle the data sparsity problem is active learning [19] [117], in order to acquire high quality data by querying users to rate a given set of items. The goal is to get the maximized error reduction with the least queries for users since obtaining information from them is costly. To achieve this purpose, the system requests the user to rate specific items based on certain criteria, a.k.a. active learning strategies. In Chapter 4, an active learning framework is proposed, which incorporates the traditional user-focused active learning with item-focused active learning, to improve the performance of the whole system. However, this proposed work and existing research on active learning [120] [131] only applies and evaluates elicitation strategies on a single-domain scenario.

In this chapter, a novel multi-domain active learning framework is proposed, which combines active learning with the cross-domain collaborative filtering algorithm in the multi-domain scenarios. A variety of elicitation strategies are evaluated on the proposed multi-domain active learning framework which elicits ratings based on the criteria with regard to both items and users, for the purpose of improving the performance of the whole system, and in which *Rating-Matrix Generative Model (R-MGM)* is employed as the cross-domain algorithm that collectively learns different systems simultaneously. The experiments are carried out among *Movielens*, *Netflix* and *Book-Crossing* datasets. The results show that the system performance can be improved further when combining cross-domain collaborative filtering with active learning algorithms.

5.2 Related Work

The proposed work is related to the emerging topic of cross-domain collaborative filtering. Most existing cross-domain collaborative filtering algorithms require the overlap between users or items, and try to transfer or aggregate knowledge by merging user preferences [114], or by mediating user modeling data [138] [139], or by combining recommendations [141], or by linking domains [143] [144], or by sharing latent features [139] [151]. In contrast, another group of algorithms focused on transferring rating patterns, such as *CBT* [152] and *RMGM* [155] where no overlap between users or items is needed. *CBT* is an adaptive method that allows knowledge transferring from the auxiliary domain to the target domain, by building the codebook as a bridge. Unlike *CBT* that builds the codebook on a dense auxiliary domain data, *RMGM* aggregates all the rating matrices in different domains to extract the shared rating patterns. *RMGM* can be seen as the probabilistic version of *CBT* for multi-task learning. Therefore, *RMGM* is employed in this work as the cross-domain collaborative filtering algorithm for evaluating different active learning strategies in multi-domain scenario.

Since the elicitation process is applied in the proposed framework, the proposed work is also related to active learning. Specifically, this work evaluates different active learning strategies proposed by [20] [117] [120]. In the review work of [121], Elahi et al. summarized all the elicitation strategies and classified them as personalized or non-personalized. Elahi et al. [131] proposed that the rating elicitations of users not only improve the prediction of the target user but also help the system to give suggestions for other users. In chapter 4 a generalized system-driven active learning framework is proposed by incorporating the user-focused with item-focused active learning strategies. However, all the previous work focused on querying ratings from a single-domain, while in this proposed work a more complicated scenario is considered by introducing more domains.

The work of combining active learning and cross-domain collaborative filtering is quite limited. In the work of [163], Zhao et al. extended previous transfer learning approaches in a partial entity-corresponding manner and proposed several entity selection strategies to actively construct entity-correspondences across different recommender systems. In their method, the proposed rating elicitation strategies are based a specific model where partial-correspondence is needed. Although the cross-domain entity-correspondences are unknown, the mappings between domains need to be identified at a cost. While in the proposed method a selection of active learning strategies are evaluated based on the *RMGM* model where no correspondence is needed. Another major difference is that their algorithm is based on the cross-domain scenario while this work tries to solve the multi-domain recommendation problem. Zhang et al. [164] proposed an active learning strategy for multi-domain recommendation based on the global generalization error. For each rating in the learning set, they estimate the global generalization error as the aggregation of the generalization error in domain-specific knowledge and the generalization error in the domain-independent knowledge. Only the ratings with the least global generalization error are elicited. Their work is based on the assumption that the ratings in the learning set are known, which may not hold. While in the proposed framework such assumption is not made, i.e. the ratings requested are not the same as the ratings acquired. In addition, their active learning strategy elicit only one rating per request, while this work assumes that the system makes many rating requests at the same time. Last but not least, they compare the proposed approach only with the random strategy, while this work studies the performance of several strategies.

5.3 *Rating-Matrix Generative Model (RMGM)*

This section reviews the *RMGM* model, which is a cross-domain collaborative filtering algorithm that allows knowledge-sharing across multiple rating matrices [155]. Given a set of rating matrices in related domains $R = \{R_{(1)}, ..., R_{(D)}\}$ ($R_{(t)} \in \mathbb{R}^{m_t \times n_t}$), in which the user set is denoted as $I_d = \{i_1^{(d)}, ..., i_{n_d}^{(d)}\}$, the item set is denoted as $J_d = \{j_1^{(d)}, ..., j_{m_d}^{(d)}\}$ and the rating data is denoted as $R_d = \{(i_1^{(d)}, j_1^{(d)}, r_1^{(d)}), ..., (i_{s_d}^{(d)}, j_{s_d}^{(d)}, r_{s_d}^{(d)})\}$ in the d-th domain. The task of is to learn a *RMGM* for the given related tasks on the pooled rating data and predict missing values in all domains.

RMGM assumes that users/items can simultaneously belong to multiple clusters since users may have multiple personalities and items may have multiple attributes. *RMGM* establishes a cluster-level rating-pattern representation as a 'bridge' to connect all the domains, based on the assumption that latent correlations may exist between preferences of group of users for group of items (such as users' interests for item genre). Suppose there are K user clusters $\{c_{\mathcal{U}}^{(1)}, ..., c_{\mathcal{U}}^{(K)}\}$ and L item clusters $\{c_{\mathcal{V}}^{(1)}, ..., c_{\mathcal{V}}^{(L)}\}$ in the cluster-level rating patterns, the marginal distributions for user i and item j are:

$$P_{\mathcal{U}}(i) = \sum_k P(c_{\mathcal{U}}^{(k)}) P(i|c_{\mathcal{U}}^{(k)}) \tag{5.1}$$

$$P_{\mathcal{V}}(j) = \sum_l P(c_{\mathcal{V}}^{(l)}) P(j|c_{\mathcal{V}}^{(l)}) \tag{5.2}$$

Then the ratings can be drawn from the user and the item mixture models (*User-Item Joint Mixture Model*):

$$(i_t^{(d)}, j_t^{(d)}) \sim \sum_{kl} P(c_{\mathcal{U}}^{(k)}) P(c_{\mathcal{V}}^{(l)}) P(i|c_{\mathcal{U}}^{(k)}) P(j|c_{\mathcal{V}}^{(l)}) \tag{5.3}$$

In addition, the ratings also can be drawn from the conditional distributions

given the latent cluster variables (*Cluster-Level Rating Model*):

$$r_t^{(d)} \sim P(r|c_{\mathcal{U}}^{(k)}, c_{\mathcal{V}}^{(l)}) \tag{5.4}$$

Combining Equation 5.3 and 5.4 gives *Rating Matrix Generative Model (RMGM).*

For training the *RMGM*, five sets of parameters in *RMGM* need to be learnt: $P(c_{\mathcal{U}}^{(k)}), P(c_{\mathcal{V}}^{(l)}), P(i|c_{\mathcal{U}}^{(k)}), P(j|c_{\mathcal{V}}^{(l)})$, and $P(r|c_{\mathcal{U}}^{(k)}, c_{\mathcal{V}}^{(l)})$ (for $k = 1, ..., K; l = 1, ..., L; i \in \cup_d \mathcal{U}_d; j \in \cup_d \mathcal{V}_d, r \in R$).

Expectation Maximization (EM) algorithm is adopted for *RMGM* training. Specifically, in the E-step: the joint posterior probability $P(c_{\mathcal{U}}^{(k)}, c_{\mathcal{V}}^{(l)}|i_t^{(d)}, j_t^{(d)}, r_t^{(d)})$ is computed using the five sets of parameters. In the M-step: the five sets of parameters for D given tasks are updated based on $P(c_{\mathcal{U}}^{(k)}, c_{\mathcal{V}}^{(l)}|i_t^{(d)}, j_t^{(d)}, r_t^{(d)})$. By alternating E-step and M-step, an *RMGM* model which fits the given multiple tasks can be obtained.

To predict missing values for an existing user, the rating function can be generated by:

$$\begin{aligned} f_R(i_t^{(d)}, j_t^{(d)}) &= \sum_r r P(r|i_t^{(d)}, j_t^{(d)}) \\ &= \sum_r r \sum_{kl} P(r|c_{\mathcal{U}}^{(k)}, c_{\mathcal{V}}^{(l)}) P(c_{\mathcal{U}}^{(k)}, c_{\mathcal{V}}^{(l)}|i_t^{(d)}, j_t^{(d)}) \\ &= \sum_r r \sum_{kl} P(r|c_{\mathcal{U}}^{(k)}, c_{\mathcal{V}}^{(l)}) P(c_{\mathcal{U}}^{(k)}|i_t^{(d)}) P(c_{\mathcal{V}}^{(l)}|j_t^{(d)}) \end{aligned} \tag{5.5}$$

where $P(c_{\mathcal{U}}^{(k)}|i_t^{(d)})$ and $P(c_{\mathcal{V}}^{(l)}|j_t^{(d)})$ can be computed using the learned parameters based on Bayes rule.

To predict the ratings for a new user, a quadratic optimization problem can be solved to estimate the user-cluster membership $\mathbf{p}_{i^{(d)}} \in \mathbb{R}^K$ for $i^{(d)}$ based on the

given ratings $\mathbf{r}_{i^{(d)}}$:

$$\min_{\mathbf{p}_{i^{(d)}}} \left\| [\mathbf{B}\mathbf{P}_{J_d}]^T \mathbf{p}_{i^{(d)}} - \mathbf{r}_{i^{(d)}} \right\|^2_{\mathbf{W}_{i^{(d)}}}, s.t. \mathbf{P}_{i^{(d)}} \mathbf{1} = 1 \tag{5.6}$$

where $\mathbf{B}_{kl} = \sum_r rP(r|c_{\mathcal{U}}^{(k)}, c_{\mathcal{V}}^{(l)})$, $[\mathbf{P}_{J_d}]_{lt} = P(c_{\mathcal{V}}^{(l)}|j_t^{(d)})$, $[\mathbf{W}_{i^{(d)}}]_{tt} = 1$ if $[r_{i^{(d)}}]_t$ is given, $[\mathbf{W}_{i^{(d)}}]_{tt} = 0$ otherwise. After obtaining the optimal user-cluster membership $\tilde{\mathbf{p}}_{i^{(d)}}$ for $i^{(d)}$, the ratings of user $i^{(d)}$ on item $j_t^{(d)}$ can be predicted by:

$$f_R(i_t^{(d)}, j_t^{(d)}) = \tilde{\mathbf{p}}_{i^{(d)}}^T \mathbf{B} \mathbf{p}_{j_t^{(d)}} \tag{5.7}$$

where $\mathbf{p}_{j_t^{(d)}}$ is the t-th column in P_{J_d}. Alternatively, the ratings of all the existing users on a new item can be predicted in the similar way. Overall, all the missing ratings among all related domains can be obtained by *RMGM*.

5.4 Active Learning for Multi-Domain Recommendations

Traditional active learning [120] [128] [129] is a set of techniques that intelligently elicit ratings for users when a new user comes in. These researches only evaluate each user independently and only consider the benefits of the elicitations to new users, but pay less attention to the effects of the system. In Chapter 4 a novel system-driven active learning framework is proposed for improving the performance of the whole system. A multi-domain algorithm utilizes the shared knowledge across multiple domains to alleviate the data sparsity in all domains, in order to improve the performance of the whole systems in all domains. Therefore, both active learning and multi-domain collaborative filtering algorithm aim at improving the performance of the systems when the active learning is considered in multi-domain scenarios.

Based on this assumption, a novel multi-domain active learning framework is proposed by incorporating active learning with multi-domain collaborative filtering

Algorithm 5.1 Multi-Domain Active Learning Framework

Input: Training set κ which is collected from D domains that have been assigned values; learning set ϕ $(\times D)$ that are known by the users in D domains; test set τ which consists of a number of ratings in D domains that are supposed to be predicted by the system; Predefined iteration time K;

Output: Evaluation (often measured by RMSE, MAE, etc.) of the test set τ from D domains;

In each iteration:

Step 1: Select a set of ratings $\chi_1 \in \tau$ $(\times D)$ based on a predefined item selection criterion (active learning strategy) where $\chi_1 = \{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(D)}\}$;

Step 2: Select a set of ratings $\chi_2 \in \tau$ $(\times D)$ based on a predefined user selection criterion (active learning strategy) where $\chi_2 = \{\mathbf{y}^{(1)}, ..., \mathbf{y}^{(D)}\}$;

Step 3: Only the ratings that are both selected from Step 1 and Step 2 are considered as elicitations, in this case $\chi_3 = \chi_1 \cap \chi_2$, therefore $\chi_3 = \{\mathbf{x}^{(1)} \cap \mathbf{x}^{(1)}, ..., \mathbf{x}^{(D)} \cap \mathbf{x}^{(D)}\})$;

Step 4: Add the selected rating (or ratings) from Step 3 into the training set κ, therefore $\kappa = \{\kappa, \chi_3\}$;

Step 6: Remove the selected ratings χ_3 from the learning set ϕ $(\times D)$;

Step 7: Train the *RMGM* based on the updated training set κ;

Step 8: Evaluate the predictions in test set τ based on the trained prediction model.

Step 9: Repeat Step 1 to Step 8 for K times;

as shown in Algorithm 5.1. Specifically, in multi-domain scenarios, the model is learnt by aggregating data from all domains, which is referred as the training set. One of the advantages of the *RMGM* is that no overlap between items or users is needed, meaning that no connection of users or items need to be built between domains. As a result, each elicited rating contributes not only to the domain it belongs to, but also have an effect on other ones. Thus, active learning can be utilized for the single-domain, or for multi-domains, i.e. the ratings could be elicited from the learning set that contains ratings from a single dataset, or from all the datasets. As mentioned in the proposed active learning framework (in Chapter 4), in each iteration, only users who fulfill the user selection strategy are queried for ratings only on the items which satisfy the item selection strategy. As a result, the intersections of the elicited ratings that are both selected based on the item selection criterion and user selection criterion are elicited from the learning set to

the training set (Step 1 to Step 6). The task of *RMGM* is to alleviate the sparsity in all domains. Therefore, the model is tested on the dataset (test set) which consists of ratings from all the datasets.

In the next section, comprehensive evaluations are given for demonstrating the effectiveness of the proposed framework.

5.5 Evaluations of the Proposed Framework

5.5.1 Datasets and Experimental Setup

As similar to the work of [155], three real-world collaborative filtering datasets are used for performance evaluation: *Movielens*, *Netflix* and *Book-Crossing*.

Active learning strategies are evaluated on proposed framework in which the shared model *(RMGM)* is built on the union of the rating data from these three dataset. All the known ratings are partitioned randomly into three sets for all three domains:

- Training set: contains 20% of the ratings, which are considered as known by the system. The ratings in this dataset are used for training the matrix factorization model in each iteration in the active learning process.

- Learning set: contains 60% of the ratings, which are regarded as known by the users but not known by the system. Therefore, the ratings in this dataset are elicited incrementally to the training set if the system queries.

- Test set: contains 20% of the ratings that are used to evaluate the elicitation strategies.

In the experiments the number of queried items and users are set to be from 0% to 100% with 1% increase (simply setting items equal users in percentage) in each iteration based on different strategies. Therefore, the number of iterations is 101 from the stage of training the model with no elicitation to the stage with all

the elicitations. Then all the ratings in the learning set are elicited incrementally to the training set, which is utilized for building the *RMGM*. Each *RMGM* follows the same preprocessing step of [155]: the number of latent user groups shared across domains is 20, the number of latent item groups shared across domains is 20, while the *EM* algorithm iteration number is 50.

At last, the performance of system is evaluated by comparing the difference between the predictions from the model (RMGM) and the ground truth in test set, usually measured by their RMSE:

$$\text{RMSE} = \sqrt{\frac{\sum\limits_{(i,j \in TestSet)} (r_{ij} - \tilde{r}_{ij})^2}{T}} \tag{5.8}$$

where r_{ij} is the rating that the ith user gives to the jth item, $\tilde{r}_{ij}$ is the predicted rating accordingly, and T is the total number of test samples in all domains.

These settings are based on the assumption that all three recommender systems were first built by small datasets (20%). Afterwards they keep obtaining ratings by querying different users about different items based on corresponding strategies. By iteratively acquiring knowledge from elicitations (from 20% to 80%), the systems could generate more and more precise recommendations and have effect on other ones.

5.5.2 Evaluation Strategies

A variety of active learning strategies (a.k.a. elicitation criteria) are evaluated based on the multi-domain active learning framework: *Random*, *Frequency*, *Variance*, *Entropy*, *Entropy0*, *Log(Frequency)*Entropy*, *Sqrt(Frequency)*Variance*, *Coverage* and *HELF*. The details of these strategies can be found in Section 4.3.2.

In addition, four cases are studied by utilizing active learning in different source domain(s): elicit ratings from each single-domain *Movielens*, *Netflix* and *Book-Crossing* or multiple domains *Movielens+Netflix+Book-Crossing*.

5.5.3 Performance Analyses

In the proposed active learning framework, ratings are elicited by asking a set of users about certain items iteratively through batch learning (readjust the model after eliciting several ratings). Elicitations must take into consideration that users are willing or not to answer such queries. For example, if an user has not watched queried movies, he or she is not able to provide the rating for this movie. Therefore, only the ratings known by the user (in the learning set) are elicited. The number of acquired ratings represents the ability of active learning strategy to estimate what item the user has actually experienced and is therefore able to rate. However, this measure is based on the knowledge of each single-domain, which is not considered in this work.

This section presents the results of the experiments based on two aspects: system *RMSE* evolution in the learning process and the quality of elicited ratings.

5.5.3.1 *RMSE* - Iteration

Similar to the work of Chapter 4, ratings are elicited by asking a set of users about certain items iteratively through batch learning based on the proposed framework. In multi-domain scenario, active learning can be applied in each single-domain or through multiple domains. The performance of all the strategies are first presented in terms of prediction error (*RMSE*) in multi-domain scenario versus the proportion of queried items and users in each single-domain and multiple domains, which models the learning process.

Figure 5.1 to 5.4 depict how the *RMSE* of multi-domain recommender system (*Netflix+Movielens+Book-Crossing*) varies as rating elicitations are acquired from *Movielens*, *Netflix*, *Book-Crossing*, and all three datasets, respectively.

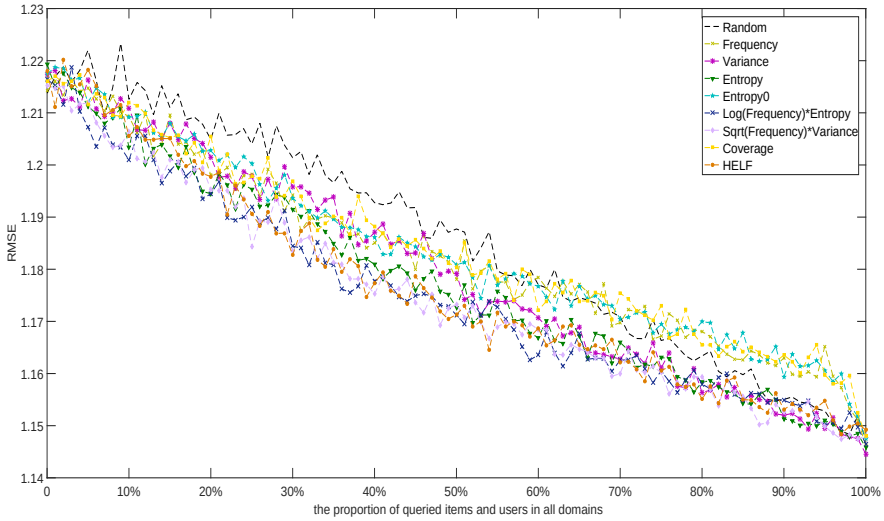


Figure 5.1: System RMSE evolution on *Netflix+Movielens+Book-Crossing* with elicitations from *Movielens*

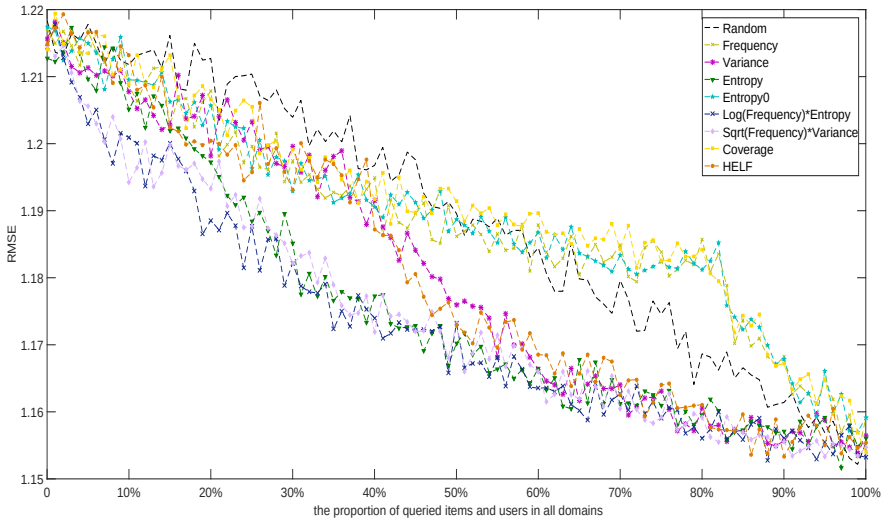


Figure 5.2: System RMSE evolution on *Netflix+Movielens+Book-Crossing* with elicitations from *Netflix*

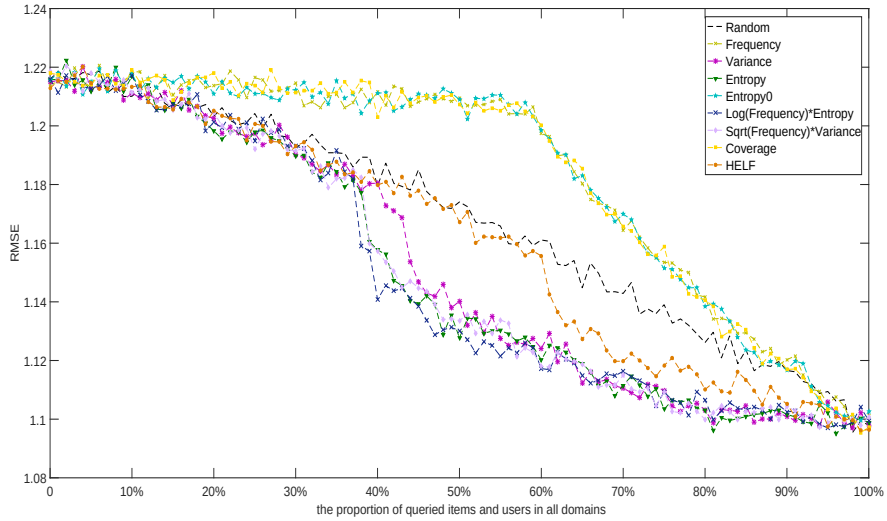


Figure 5.3: System RMSE evolution on *Netflix+Movielens+Book-Crossing* with elicitations from *Book-Crossing*

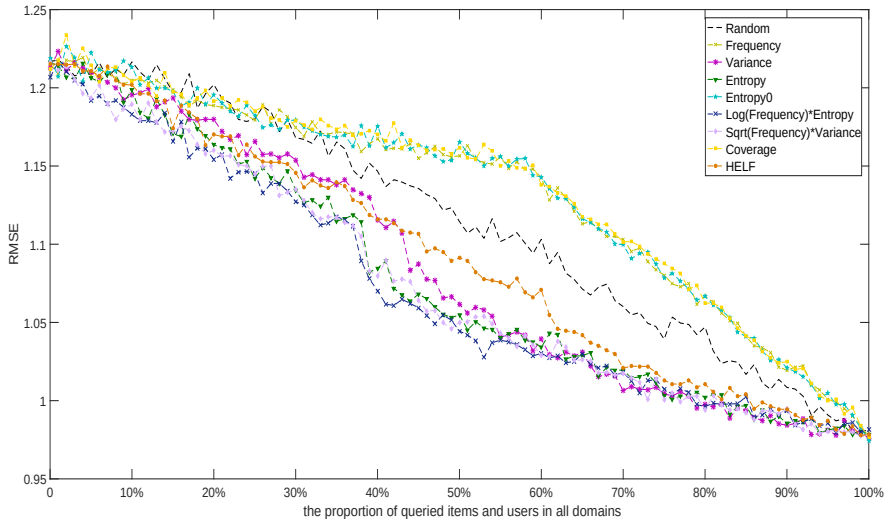


Figure 5.4: System RMSE evolution on *Netflix+Movielens+Book-Crossing* with elicitations from all three datasets

Specifically, the performing difference between strategies is very limited during the learning process for the case of utilizing active learning only on the *Movielens* dataset (as shown in Figure 5.1). The *Sqrt(Frequency)*Variance* strategy is slightly better than other strategies, while the *Frequency*, *Entropy0* and *Coverage* strategies generate poorer prediction accuracy than others. For the experiments in which active learning is applied on the *Netflix* and *Book-Crossing* dataset (Figure 5.2 and Figure 5.3, respectively), the difference between strategies is obvious: the *Variance*,*Entropy*, *Log(Frequency)*Entropy*, *Sqrt(Frequency)*Variance* strategies produce lower *RMSE* than the *Random* strategy, while the *Frequency*, *Entropy0* and *Coverage* strategies are on the opposite.

Since the tendencies of different strategies in each single-domain (*Movielens*, *Netflix* and *Book-Crossing*) are similar, requesting ratings from all three datasets will lead to the same results (as shown in Figure 5.4).

Overall, it is apparent that the performance of multi-domain recommender system is improved by rating elicitations through active learning techniques (lower the *RMSE* from the starting point to the end point).

5.5.3.2 RMSE - Number of Elicited Ratings

In real-life scenarios, the users are often reluctant to give ratings when the system queries frequently, which is against the experimental assumptions. In addition, the labeling work is costly since it requires human effort, sometimes even lower the users satisfaction, the quality of the elicited ratings is the priority to be concerned. Thus, the performance of all the strategies in terms of prediction error (*RMSE*) in multi-domain scenario versus the number of elicited ratings in each single-domain and multiple domains are reported.

As shown from Figure 5.5 to Figure 5.7, the ratings acquired from each single-domain by the *Frequency*, *Entropy0* and *Coverage* strategies generate poor performance than the *Random* strategy. Ratings elicited based on the *Variance*,*Entropy*,

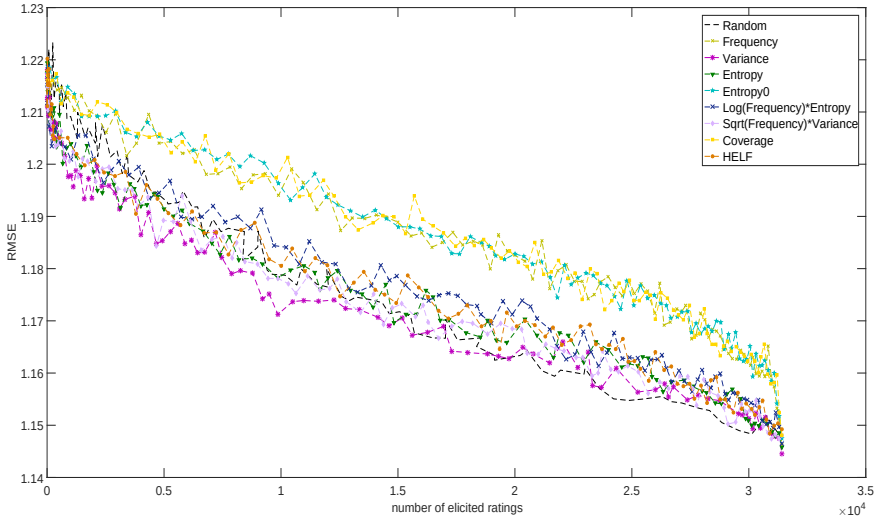


Figure 5.5: System RMSE evolution on *Netflix+Movielens+Book-Crossing* versus the number of elicited ratings from *Movielens*

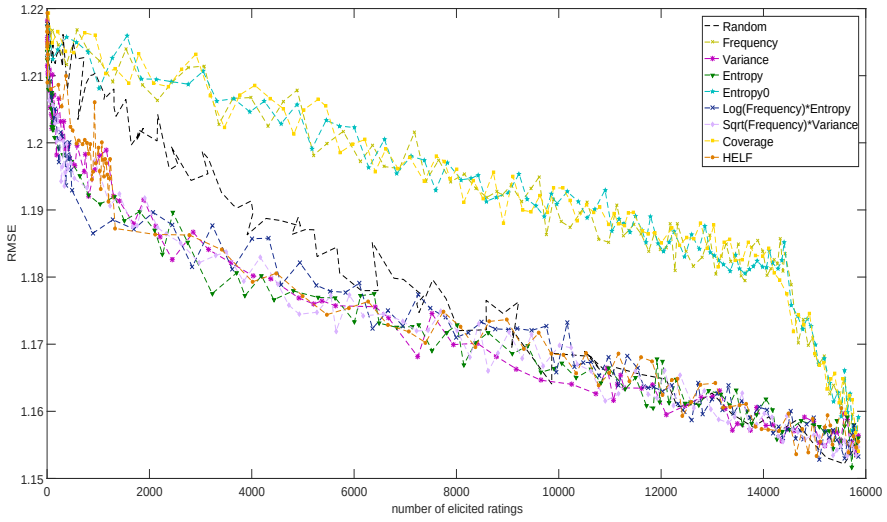


Figure 5.6: System RMSE evolution on *Netflix+Movielens+Book-Crossing* versus the number of elicited ratings from *Netflix*

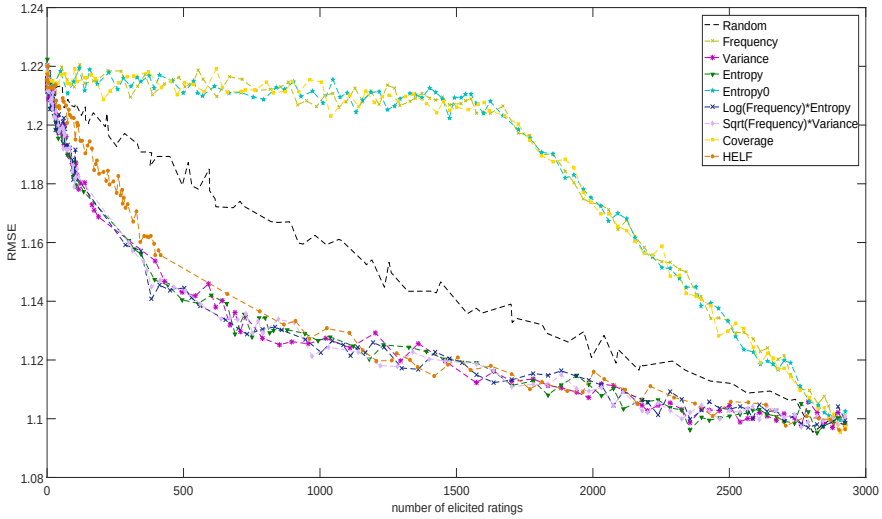


Figure 5.7: System RMSE evolution on *Netflix+Movielens+Book-Crossing* versus the number of elicited ratings from *Book-Crossing*

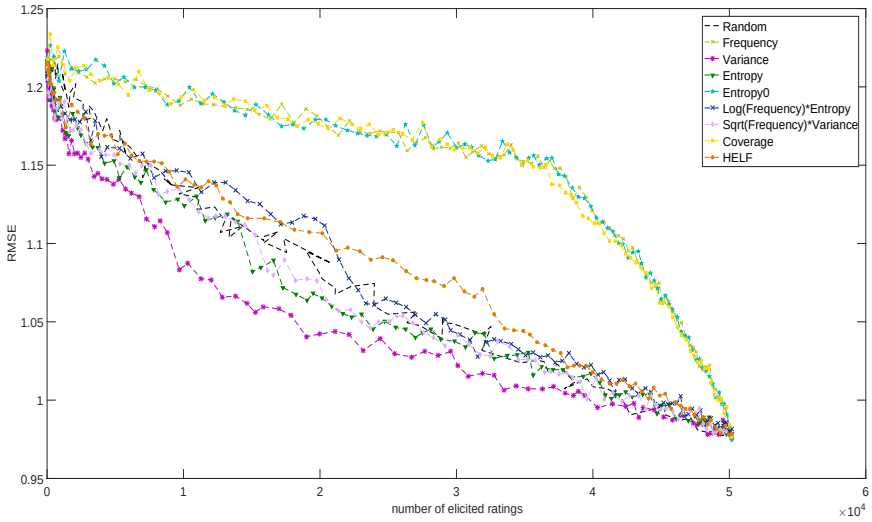


Figure 5.8: System RMSE evolution on *Netflix+Movielens+Book-Crossing* versus the number of elicited ratings from all three datasets

*Log(Frequency)*Entropy*, *Sqrt(Frequency)*Variance* strategies could obtain lower *RMSE*. Aggregating all the elicitation from single-domain, the similar results are obtained (as shown in Figure 5.8).

The *Frequency*, *Entropy0* and *Coverage* strategies tend to select users and items with a large number of ratings, which may contain bias that deteriorates the cluster-level rating-pattern in *RMGM* by misleading the user and item into inaccurate clusters. However, the *Variance,Entropy*, *Log(Frequency)*Entropy*, *Sqrt(Frequency)*Variance* strategies produce promising results by incorporating informative and diverse ratings, which are considered to be suitable for the case of multi-domain recommender systems in the framework of *RMGM*.

5.6 Summary

This chapter introduces a new multi-domain active learning framework which incorporates the proposed active learning framework with the cross-domain collaborative filtering algorithm (*RMGM*) for the multi-domain recommendations, in order to alleviate the sparsity problem in all domains. Furthermore, several widely used active learning strategies are applied and evaluated on the proposed framework with various elicitation sources (from each single-domain and multi-domains). The experimental results has shown that the elicitations from different source domain (domains) generate similar results. That means the performance of multi-domain system is insensitive to the choice of elicitation sources, but rely on the characteristics of the elicitations. More importantly, it shows that incorporating active learning techniques can further improve the multi-domain recommender systems, especially for the *Variance,Entropy*, *Log(Frequency)*Entropy*, *Sqrt(Frequency)*Variance* strategies.

Chapter 6

Conclusion

6.1 Thesis Summary

The lack of information is an acute challenge in most recommender systems, especially for the collaborative filtering algorithms which utilize user-item rating matrix (matrices) as the only source of information. In this thesis, the sparsity problem of collaborative filtering recommender systems have been addressed in three directions: automatically 'add' ratings learnt by the system with collaborative filtering algorithms; manually add ratings by requesting users through active learning techniques; exploit knowledges from other domains with cross-domain collaborative filtering methods for reducing the sparsity of the target domain(s).

6.2 Contribution

The contribution of this thesis is summarized as follow.

- In Chapter 3, a new matrix factorization model called *Enhanced SVD (ESVD)* is proposed, which combines the classic matrix factorization algorithms with ratings completion inspired by active learning. Then it shows that this general framework can be incorporated with different *SVD-based* algorithms such

as *SVD++* by proposing the *ESVD++* method. In addition, the connection between the prediction accuracy and the density of matrix is built to further explore its potentials. Based on this theory, the *Multi-layer ESVD (MESVD)* is introduced, which learns the model iteratively to further improve the prediction accuracy. This *MESVD* approach can achieve better performance than *ESVD* but in sacrifice of training time. To handle the imbalanced datasets that contain far more users than items or more items than users, the *Item-wise ESVD (IESVD)* and *User-wise ESVD (UESVD)* are presented, respectively. Experimental results suggest their effectiveness in terms of accuracy when compared with traditional matrix factorization methods. Furthermore, this ratings completion strategy tackles the problem of active learning of which the requesting process is costly and unrealistic.

- In Chapter 4, a novel generalized framework for applying active learning in recommender systems is proposed. In the proposed framework, the ratings are elicited simultaneously based on the criteria with regard to both items and users, for the purpose of improving the performance of the whole system. The evaluations have shown that different strategies can improve different aspects of the recommender system in different stages for different datasets. The experimental results have shown that the proposed framework could achieve better performance based on certain strategies by taking extra information (users' ratings) into consideration, when compared with the conventional active learning.

- In Chapter 5, a novel multi-domain active learning framework is proposed, which incorporates the former proposed active learning framework (in Chapter 4) with the cross-domain collaborative filtering algorithm (*RMGM*) for the multi-domain recommendations, in order to alleviate the sparsity problem in all domains. Several widely used active learning strategies are applied and e-

valuated on the proposed framework with various elicitation sources (from each single-domain and multi-domains). The *Variance*,*Entropy*, *Log(Frequency)*Entropy*, *Sqrt(Frequency)*Variance* strategies are proved to be effective for tackling the active learning task in the *RMGM* model based on the dataset (*Movielens+Netflix+Book-Crossing*) from three different domains. The experimental results also show that incorporating active learning techniques can further improve the multi-domain recommender systems.

6.3 Future Work

In this thesis, some works have been done for addressing the sparsity problem of the collaborative filtering recommender systems, while more works are yet to be done in order to further improve them. Here some possible new lines of investigation for future research are listed.

- In Chapter 3, only the *SVD-based* algorithms are implemented on the extracted sub-matrix, which can be seen as an independent recommender system. Based on the same idea, other collaborative filtering algorithms can be tested according to the characteristics of the extracted sub-matrix for the better prediction accuracy of the pre-estimations. A possibility is to employ memory-based algorithms for the pre-estimations in the sub-matrix since memory-based algorithms achieve good performance when the rating matrix is dense.

- In Chapter 4, the experiments are conducted based on the assumption that users would only give ratings when the system queries, without considering the ratings that users voluntarily rate. Moreover, in this work only the pure strategies that are applicable to both items and users are studied, while mixed strategies including item-specific and user-specific strategies still remain further explorations. Last but not least, sequentially applying different strategies

with different prediction algorithms in different stages is also a possibility for generating better performance.

- In Chapter 5, active learning strategies are utilized for eliciting ratings from the Movie domain and Book domain, and then an initial conclusion is obtained: the performance of multi-domain system is less insensitive to the choice of elicitation sources, but more rely on the characteristics of the elicitations. However, sources from other domains still worth further exploration since Movie domain and Book domain have a lot in common. In addition, the combination of *Movielens+Netflix+Book-Crossing* is used for training the *RMGM* model and evaluating the proposed multi-domain active learning method, which can only be considered as one dataset. In the future the proposed algorithm will be tested on more datasets from various domains. Last but not least, only the active learning in multi-domain scenarios is considered in where the shared knowledge across multiple domains is utilized to alleviate the sparsity in all domains. In the future the feasibility of applying the proposed active learning in cross-domain scenarios will be studied in where the knowledge derived from the source domain is utilized to alleviate the sparsity in the target domain.

Bibliography

[1] J. Bennett, S. Lanning *et al.*, "The netflix prize," in *Proceedings of KDD cup and workshop*, vol. 2007. New York, NY, USA, 2007, p. 35.

[2] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 271–280.

[3] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," vol. 7, no. 1. IEEE, 2003, pp. 76–80.

[4] E.-A. Baatarjav, S. Phithakkitnukoon, and R. Dantu, "Group recommendation system for facebook," in *On the Move to Meaningful Internet Systems: OTM 2008 Workshops*. Springer, 2008, pp. 211–219.

[5] X. Amatriain, A. Jaimes, N. Oliver, and J. M. Pujol, "Data mining methods for recommender systems," in *Recommender Systems Handbook*. Springer, 2011, pp. 39–71.

[6] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 5–53, 2004.

[7] P. Melville, R. J. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," in *Aaai/iaai*, 2002, pp. 187–192.

[8] J. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," *The adaptive web*, pp. 291–324, 2007.

[9] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[10] P. Lops, M. De Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in *Recommender systems handbook*. Springer, 2011, pp. 73–105.

[11] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno, "Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences." in *ISMIR*, vol. 6, 2006, p. 7th.

[12] S. Trewin, "Knowledge-based recommender systems," *Encyclopedia of library and information science*, vol. 69, no. Supplement 32, p. 180, 2000.

[13] D. Bridge, M. H. Göker, L. McGinty, and B. Smyth, "Case-based recommender systems," *The Knowledge Engineering Review*, vol. 20, no. 03, pp. 315–320, 2005.

[14] R. Burke, "The wasabi personal shopper: a case-based recommender system," in *AAAI/IAAI*, 1999, pp. 844–849.

[15] A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker, "Developing constraint-based recommenders," in *Recommender systems handbook*. Springer, 2011, pp. 187–215.

[16] A. Felfernig and A. Kiener, "Knowledge-based interactive selling of financial services with fsadvisor," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 20, no. 3. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005, p. 1475.

[17] A. García-Crespo, J. Chamizo, I. Rivera, M. Mencke, R. Colomo-Palacios, and J. M. Gómez-Berbís, "Speta: Social pervasive e-tourism advisor," *Telematics and Informatics*, vol. 26, no. 3, pp. 306–315, 2009.

[18] R. Burke, "Hybrid recommender systems: Survey and experiments," *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[19] N. Rubens, M. Elahi, M. Sugiyama, and D. Kaplan, "Active learning in recommender systems," in *Recommender systems handbook*. Springer, 2015, pp. 809–846.

[20] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl, "Getting to know you: learning new user preferences in recommender systems," in *Proceedings of the 7th international conference on Intelligent user interfaces*. ACM, 2002, pp. 127–134.

[21] G. Carenini, J. Smith, and D. Poole, "Towards more conversational and collaborative recommender systems," in *Proceedings of the 8th international conference on Intelligent user interfaces*. ACM, 2003, pp. 12–18.

[22] M. Elahi, F. Ricci, and N. Rubens, "Active learning in collaborative filtering recommender systems," in *International Conference on Electronic Commerce and Web Technologies*. Springer, 2014, pp. 113–124.

[23] I. Cantador, I. Fernández-Tobías, S. Berkovsky, and P. Cremonesi, "Cross-domain recommender systems," in *Recommender Systems Handbook*. Springer, 2015, pp. 919–959.

[24] Y. Zhang, B. Cao, and D.-Y. Yeung, "Multi-domain collaborative filtering," in *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2010, pp. 725–732.

[25] B. N. Miller, I. Albert, S. K. Lam, J. A. Konstan, and J. Riedl, "Movielens unplugged: experiences with an occasionally connected recommender system," in *Proceedings of the 8th international conference on Intelligent user interfaces*, 2003, pp. 263–266.

[26] P. McJones, "Eachmovie collaborative filtering data set," *DEC Systems Research Center*, vol. 249, 1997.

[27] C.-N. Ziegler and D. Freiburg, "Book-crossing dataset," 2004.

[28] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *information retrieval*, vol. 4, no. 2, pp. 133–151, 2001.

[29] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer, "The yahoo! music dataset and kdd-cup'11." in *KDD Cup*, 2012, pp. 8–18.

[30] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1998, pp. 43–52.

[31] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Analysis of recommendation algorithms for e-commerce," in *Proceedings of the 2nd ACM conference on Electronic commerce*. ACM, 2000, pp. 158–167.

[32] C. D. Manning, P. Raghavan, H. Schütze *et al.*, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1, no. 1.

[33] G. Shani and A. Gunawardana, "Evaluating recommendation systems," in *Recommender systems handbook*. Springer, 2011, pp. 257–297.

[34] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE transactions on knowledge and data engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[35] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1.

[36] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[37] W. Hill, L. Stead, M. Rosenstein, and G. Furnas, "Recommending and evaluating choices in a virtual community of use," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press/Addison-Wesley Publishing Co., 1995, pp. 194–201.

[38] U. Shardanand and P. Maes, "Social information filtering: algorithms for automating word of mouth," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press/Addison-Wesley Publishing Co., 1995, pp. 210–217.

[39] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "Grouplens: applying collaborative filtering to usenet news," *Communications of the ACM*, vol. 40, no. 3, pp. 77–87, 1997.

[40] J. Delgado and N. Ishii, "Memory-based weighted majority prediction," in *SIGIR Workshop Recomm. Syst. Citeseer*, 1999.

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

[42] U. Shardanand, "Social information filtering for music recommendation," Ph.D. dissertation, Citeseer, 1994.

[43] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. 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*. ACM, 1999, pp. 230–237.

[44] J. Herlocker, J. A. Konstan, and J. Riedl, "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms," *Information retrieval*, vol. 5, no. 4, pp. 287–310, 2002.

[45] P. Massa and B. Bhattacharjee, "Using trust in recommender systems: an experimental analysis," in *International Conference on Trust Management*. Springer, 2004, pp. 221–235.

[46] P. Massa and P. Avesani, "Trust-aware recommender systems," in *Proceedings of the 2007 ACM conference on Recommender systems*. ACM, 2007, pp. 17–24.

[47] M. Clements, P. Serdyukov, A. P. De Vries, and M. J. Reinders, "Using flickr geotags to predict user travel behaviour," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2010, pp. 851–852.

[48] I. Guy, N. Zwerdling, I. Ronen, D. Carmel, and E. Uziel, "Social media recommendation based on people and tags," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2010, pp. 194–201.

[49] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 285–295.

[50] R. M. Bell and Y. Koren, "Scalable collaborative filtering with jointly derived neighborhood interpolation weights," in *Data Mining, Seventh IEEE International Conference on.* IEEE, 2007, pp. 43–52.

[51] Y. 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.* ACM, 2008, pp. 426–434.

[52] M. Jamali and M. Ester, "Trustwalker: a random walk model for combining trust-based and item-based recommendation," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2009, pp. 397–406.

[53] K. H. Tso and L. Schmidt-Thieme, "Evaluation of attribute-aware recommender system algorithms on data with varying characteristics," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Springer, 2006, pp. 831–840.

[54] C. S. Firan, W. Nejdl, and R. Paiu, "The benefit of using tag-based profiles," in *Web Conference, 2007. LA-WEB 2007. Latin American.* IEEE, 2007, pp. 32–41.

[55] K. H. Tso-Sutter, L. B. Marinho, and L. Schmidt-Thieme, "Tag-aware recommender systems by fusion of collaborative filtering algorithms," in *Proceedings of the 2008 ACM symposium on Applied computing.* ACM, 2008, pp. 1995–1999.

[56] S. Sen, J. Vig, and J. Riedl, "Tagommenders: connecting users to items through tags," in *Proceedings of the 18th international conference on World wide web.* ACM, 2009, pp. 671–680.

[57] H. Liang, Y. Xu, Y. Li, R. Nayak, and X. Tao, "Connecting users and items with weighted tags for personalized item recommendations," in *Proceedings of the 21st ACM conference on Hypertext and hypermedia.* ACM, 2010, pp. 51–60.

[58] Y. Takeuchi and M. Sugimoto, "Cityvoyager: an outdoor recommendation system based on user location history," in *International Conference on Ubiquitous Intelligence and Computing.* Springer, 2006, pp. 625–636.

[59] T. Horozov, N. Narasimhan, and V. Vasudevan, "Using location for personalized poi recommendations in mobile environments," in *Applications and the internet, 2006. SAINT 2006. International symposium on.* IEEE, 2006, pp. 6–pp.

[60] D. Billsus and M. J. Pazzani, "Learning collaborative information filters." in *Icml*, vol. 98, 1998, pp. 46–54.

[61] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender system-a case study," DTIC Document, Tech. Rep., 2000.

[62] L. H. Ungar and D. P. Foster, "Clustering methods for collaborative filtering," in *AAAI workshop on recommendation systems*, vol. 1, 1998, pp. 114–129.

[63] A. K.-B. Merialdo, "Clustering for collaborative filtering applications," *Intelligent Image Processing, Data Analysis & Information Retrieval*, vol. 3, p. 199, 1999.

[64] C. C. Aggarwal, J. L. Wolf, K.-L. Wu, and P. S. Yu, "Horting hatches an egg: A new graph-theoretic approach to collaborative filtering," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 1999, pp. 201–212.

[65] Canny, "Collaborative filtering with privacy via factor analysis," in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2002, pp. 238–245.

[66] T. K. Landauer, P. W. Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse processes*, vol. 25, no. 2-3, pp. 259–284, 1998.

[67] T. Hofmann and J. Puzicha, "Latent class models for collaborative filtering," in *IJCAI*, vol. 99, no. 1999, 1999.

[68] T. Hofmann, "Probabilistic latent semantic analysis," in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1999, pp. 289–296.

[69] ——, "Latent semantic models for collaborative filtering," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 89–115, 2004.

[70] A. Popescul, D. M. Pennock, and S. Lawrence, "Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments," in *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2001, pp. 437–444.

[71] X. Wang, J.-T. Sun, Z. Chen, and C. Zhai, "Latent semantic analysis for multiple-type interrelated data objects," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2006, pp. 236–243.

[72] R. Wetzker, W. Umbrath, and A. Said, "A hybrid approach to item recommendation in folksonomies," in *Proceedings of the WSDM'09 Workshop on Exploiting Semantic Annotations in Information Retrieval*. ACM, 2009, pp. 25–29.

[73] T. Rattenbury and M. Naaman, "Methods for extracting place semantics from flickr tags," *ACM Transactions on the Web (TWEB)*, vol. 3, no. 1, p. 1, 2009.

[74] Z. Yin, L. Cao, J. Han, C. Zhai, and T. Huang, "Geographical topic discovery and comparison," in *Proceedings of the 20th international conference on World wide web*. ACM, 2011, pp. 247–256.

[75] L. Hong, A. Ahmed, S. Gurumurthy, A. J. Smola, and K. Tsioutsiouliklis, "Discovering geographical topics in the twitter stream," in *Proceedings of the 21st international conference on World Wide Web*. ACM, 2012, pp. 769–778.

[76] S. Funk, "Netflix update: Try this at home," 2006.

[77] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *Proceedings of KDD cup and workshop*, vol. 2007, 2007, pp. 5–8.

[78] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 791–798.

[79] A. Gunawardana and C. Meek, "A unified approach to building hybrid recommender systems," in *Proceedings of the third ACM conference on Recommender systems*. ACM, 2009, pp. 117–124.

[80] Y. Koren, "The bellkor solution to the netflix grand prize," *Netflix prize documentation*, vol. 81, pp. 1–10, 2009.

[81] Y. Ding and X. Li, "Time weight collaborative filtering," in *Proceedings of the 14th ACM international conference on Information and knowledge management*. ACM, 2005, pp. 485–492.

[82] Y. Koren, "Collaborative filtering with temporal dynamics," *Communications of the ACM*, vol. 53, no. 4, pp. 89–97, 2010.

[83] N. N. Liu, M. Zhao, E. Xiang, and Q. Yang, "Online evolutionary collaborative filtering," in *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010, pp. 95–102.

[84] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 650–658.

[85] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization." in *Nips*, vol. 1, no. 1, 2007, pp. 2–1.

[86] H. Ma, H. Yang, M. R. Lyu, and I. King, "Sorec: social recommendation using probabilistic matrix factorization," in *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008, pp. 931–940.

[87] H. Ma, M. R. Lyu, and I. King, "Learning to recommend with trust and distrust relationships," in *Proceedings of the third ACM conference on Recommender systems*. ACM, 2009, pp. 189–196.

[88] H. Ma, I. King, and M. R. Lyu, "Learning to recommend with social trust ensemble," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2009, pp. 203–210.

[89] ——, "Learning to recommend with explicit and implicit social relations," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 29, 2011.

[90] H. Ma, T. C. Zhou, M. R. Lyu, and I. King, "Improving recommender systems by incorporating social contextual information," *ACM Transactions on Information Systems (TOIS)*, vol. 29, no. 2, p. 9, 2011.

[91] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *Proceedings of the fourth ACM international conference on Web search and data mining.* ACM, 2011, pp. 287–296.

[92] Y. Zhen, W.-J. Li, and D.-Y. Yeung, "Tagicofi: tag informed collaborative filtering," in *Proceedings of the third ACM conference on Recommender systems.* ACM, 2009, pp. 69–76.

[93] Y. Shi, M. Larson, and A. Hanjalic, "Mining mood-specific movie similarity with matrix factorization for context-aware recommendation," in *Proceedings of the workshop on context-aware movie recommendation.* ACM, 2010, pp. 34–40.

[94] Y. Shi, P. Serdyukov, A. Hanjalic, and M. Larson, "Personalized landmark recommendation based on geotags from photo sharing sites." *ICWSM*, vol. 11, pp. 622–625, 2011.

[95] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang, "Collaborative location and activity recommendations with gps history data," in *Proceedings of the 19th international conference on World wide web.* ACM, 2010, pp. 1029–1038.

[96] D. Agarwal and B.-C. Chen, "Regression-based latent factor models," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2009, pp. 19–28.

[97] M. Jamali and M. Ester, "A matrix factorization technique with trust propagation for recommendation in social networks," in *Proceedings of the fourth ACM conference on Recommender systems.* ACM, 2010, pp. 135–142.

[98] D. Agarwal and B.-C. Chen, "flda: matrix factorization through latent dirichlet allocation," in *Proceedings of the third ACM international conference on Web search and data mining.* ACM, 2010, pp. 91–100.

[99] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[100] D. Agarwal, B.-C. Chen, and B. Long, "Localized factor models for multi-context recommendation," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 609–617.

[101] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[102] Y. Xu, L. Zhang, and W. Liu, "Cubic analysis of social bookmarking for personalized recommendation," in *Asia-Pacific Web Conference*. Springer, 2006, pp. 733–738.

[103] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, "Tag recommendations based on tensor dimensionality reduction," in *Proceedings of the 2008 ACM conference on Recommender systems*. ACM, 2008, pp. 43–50.

[104] ——, "A unified framework for providing recommendations in social tagging systems based on ternary semantic analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 2, pp. 179–192, 2010.

[105] S. Rendle, L. Balby Marinho, A. Nanopoulos, and L. Schmidt-Thieme, "Learning optimal ranking with tensor factorization for tag recommendation," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 727–736.

[106] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell, "Temporal collaborative filtering with bayesian probabilistic tensor factorization," in *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM, 2010, pp. 211–222.

[107] S. Moghaddam, M. Jamali, and M. Ester, "Etf: extended tensor factorization model for personalizing prediction of review helpfulness," in *Proceedings of the fifth ACM international conference on Web search and data mining.* ACM, 2012, pp. 163–172.

[108] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin, "Context-aware recommender systems." *AI Magazine*, vol. 32, no. 3, 2011.

[109] S. Rendle, "Factorization machines," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on.* IEEE, 2010, pp. 995–1000.

[110] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme, "Fast context-aware recommendations with factorization machines," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval.* ACM, 2011, pp. 635–644.

[111] S. Rendle, "Factorization machines with libfm," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 3, no. 3, p. 57, 2012.

[112] S. Rendle and L. Schmidt-Thieme, "Pairwise interaction tensor factorization for personalized tag recommendation," in *Proceedings of the third ACM international conference on Web search and data mining.* ACM, 2010, pp. 81–90.

[113] T. V. Nguyen, A. Karatzoglou, and L. Baltrunas, "Gaussian process factorization machines for context-aware recommendations," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval.* ACM, 2014, pp. 63–72.

[114] B. Loni, Y. Shi, M. Larson, and A. Hanjalic, "Cross-domain collaborative filtering with factorization machines," in *European Conference on Information Retrieval.* Springer, 2014, pp. 656–661.

[115] A. K.-B. Merialdo, "Improving collaborative filtering for new-users by smart object selection," 2001.

[116] C. Boutilier, R. S. Zemel, and B. Marlin, "Active collaborative filtering," in *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 2002, pp. 98–106.

[117] A. M. Rashid, G. Karypis, and J. Riedl, "Learning preferences of new users in recommender systems: an information theoretic approach," *ACM SIGKDD Explorations Newsletter*, vol. 10, no. 2, pp. 90–100, 2008.

[118] N. Rubens and M. Sugiyama, "Influence-based collaborative active learning," in *Proceedings of the 2007 ACM conference on Recommender systems*. ACM, 2007, pp. 145–148.

[119] C. E. Mello, M.-A. Aufaure, and G. Zimbrao, "Active learning driven by rating impact analysis," in *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010, pp. 341–344.

[120] N. Golbandi, Y. Koren, and R. Lempel, "On bootstrapping recommender systems," in *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, 2010, pp. 1805–1808.

[121] M. Elahi, V. Repsys, and F. Ricci, "Rating elicitation strategies for collaborative filtering," in *International Conference on Electronic Commerce and Web Technologies*. Springer, 2011, pp. 160–171.

[122] M. Elahi, M. Braunhofer, F. Ricci, and M. Tkalcic, "Personality-based active learning for collaborative filtering recommender systems," in *Congress of the Italian Association for Artificial Intelligence*. Springer, 2013, pp. 360–371.

[123] B. M. Marlin, R. S. Zemel, S. T. Roweis, and M. Slaney, "Recommender systems, missing data and statistical model estimation." in *IJCAI*, 2011, pp. 2686–2691.

[124] N. Golbandi, Y. Koren, and R. Lempel, "Adaptive bootstrapping of recommender systems using decision trees," in *Proceedings of the fourth ACM international conference on Web search and data mining.* ACM, 2011, pp. 595–604.

[125] K. Zhou, S.-H. Yang, and H. Zha, "Functional matrix factorizations for cold-start recommendation," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval.* ACM, 2011, pp. 315–324.

[126] N. N. Liu, X. Meng, C. Liu, and Q. Yang, "Wisdom of the better few: cold start recommendation via representative based rating elicitation," in *Proceedings of the fifth ACM conference on Recommender systems.* ACM, 2011, pp. 37–44.

[127] S. Tong and D. Koller, "Active learning for parameter estimation in bayesian networks," in *NIPS*, vol. 13, 2000, pp. 647–653.

[128] R. Jin and L. Si, "A bayesian approach toward active learning for collaborative filtering," in *Proceedings of the 20th conference on Uncertainty in artificial intelligence.* AUAI Press, 2004, pp. 278–285.

[129] A. S. Harpale and Y. Yang, "Personalized active learning for collaborative filtering," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval.* ACM, 2008, pp. 91–98.

[130] R. Karimi, C. Freudenthaler, A. Nanopoulos, and L. Schmidt-Thieme, "Non-myopic active learning for recommender systems based on matrix factoriza-

tion," in *Information Reuse and Integration (IRI), 2011 IEEE International Conference on.* IEEE, 2011, pp. 299–303.

[131] M. Elahi, F. Ricci, and N. Rubens, "Active learning strategies for rating elicitation in collaborative filtering: a system-wide perspective," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 1, p. 13, 2013.

[132] I. Fernández-Tobías, I. Cantador, M. Kaminskas, and F. Ricci, "A generic semantic-based framework for cross-domain recommendation," in *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems.* ACM, 2011, pp. 25–32.

[133] B. Li, "Cross-domain collaborative filtering: A brief survey," in *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on.* IEEE, 2011, pp. 1085–1086.

[134] I. Fernández-Tobías, I. Cantador, M. Kaminskas, and F. Ricci, "Cross-domain recommender systems: A survey of the state of the art," in *Spanish Conference on Information Retrieval*, 2012.

[135] P. Winoto and T. Tang, "If you like the devil wears prada the book, will you also enjoy the devil wears prada the movie? a study of cross-domain recommendations," *New Generation Computing*, vol. 26, no. 3, pp. 209–225, 2008.

[136] M. Nakatsuji, Y. Fujiwara, A. Tanaka, T. Uchiyama, and T. Ishida, "Recommendations over domain specific user graphs," in *Proceedings of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence.* IOS Press, 2010, pp. 607–612.

[137] S. Berkovsky, T. Kuflik, and F. Ricci, "Cross-domain mediation in collaborative filtering," in *International Conference on User Modeling.* Springer, 2007, pp. 355–359.

[138] B. Shapira, L. Rokach, and S. Freilikhman, "Facebook single and cross domain data for recommendation systems," *User Modeling and User-Adapted Interaction*, pp. 1–37, 2013.

[139] W. Pan, E. W. Xiang, and Q. Yang, "Transfer learning in collaborative filtering with uncertain ratings." in *AAAI*, vol. 12, 2012, pp. 662–668.

[140] S. Berkovsky, T. Kuflik, and F. Ricci, "Distributed collaborative filtering with domain specialization," in *Proceedings of the 2007 ACM conference on Recommender systems*. ACM, 2007, pp. 33–40.

[141] S. Givon and V. Lavrenko, "Predicting social-tags for cold start book recommendations," in *Proceedings of the third ACM conference on Recommender systems*. ACM, 2009, pp. 333–336.

[142] B. Cao, N. N. Liu, and Q. Yang, "Transfer learning for collective link prediction in multiple heterogenous domains," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 159–166.

[143] L. Getoor and C. P. Diehl, "Link mining: a survey," *ACM SIGKDD Explorations Newsletter*, vol. 7, no. 2, pp. 3–12, 2005.

[144] Y. Shi, M. Larson, and A. Hanjalic, "Tags as bridges between domains: Improving recommendation with tag-induced cross-domain collaborative filtering," in *International Conference on User Modeling, Adaptation, and Personalization*. Springer, 2011, pp. 305–316.

[145] N. Mirbakhsh and C. X. Ling, "Improving top-n recommendation for cold-start users via cross-domain information," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 9, no. 4, p. 33, 2015.

[146] W. Pan, E. W. Xiang, N. N. Liu, and Q. Yang, "Transfer learning in collaborative filtering for sparsity reduction." in *AAAI*, vol. 10, 2010, pp. 230–235.

[147] C. Ding, T. Li, W. Peng, and H. Park, "Orthogonal nonnegative matrix t-factorizations for clustering," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2006, pp. 126–135.

[148] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and C. Zhu, "Personalized recommendation via cross-domain triadic factorization," in *Proceedings of the 22nd international conference on World Wide Web.* ACM, 2013, pp. 595–606.

[149] M. Enrich, M. Braunhofer, and F. Ricci, "Cold-start management with cross-domain collaborative filtering and tags," in *International Conference on Electronic Commerce and Web Technologies.* Springer, 2013, pp. 101–112.

[150] I. Fernández-Tobías and I. Cantador, "Exploiting social tags in matrix factorization models for cross-domain collaborative filtering." in *CBRecSys@ RecSys*, 2014, pp. 34–41.

[151] T. Iwata and K. Takeuchi, "Cross-domain recommendation without shared users or items by sharing latent vector distributions." in *AISTATS*, 2015.

[152] B. Li, Q. Yang, and X. Xue, "Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction." in *IJCAI*, vol. 9, 2009, pp. 2052–2057.

[153] O. Moreno, B. Shapira, L. Rokach, and G. Shani, "Talmud: transfer learning for multiple domains," in *Proceedings of the 21st ACM international conference on Information and knowledge management.* ACM, 2012, pp. 425–434.

[154] S. Gao, H. Luo, D. Chen, S. Li, P. Gallinari, and J. Guo, "Cross-domain recommendation via cluster-level latent factor model," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases.* Springer, 2013, pp. 161–176.

[155] B. Li, Q. Yang, and X. Xue, "Transfer learning for collaborative filtering via a rating-matrix generative model," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 617–624.

[156] B. Li, X. Zhu, R. Li, C. Zhang, X. Xue, and X. Wu, "Cross-domain collaborative filtering over time," in *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*. AAAI Press, 2011, pp. 2293–2298.

[157] S. Ren, S. Gao, J. Liao, and J. Guo, "Improving cross-domain recommendation through probabilistic cluster-level latent factor model." in *AAAI*, 2015, pp. 4200–4201.

[158] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, p. 4, 2009.

[159] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 69–77.

[160] A. I. Schein, A. Popescul, L. H. Ungar, and D. 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*. ACM, 2002, pp. 253–260.

[161] C. Desrosiers and G. Karypis, "A comprehensive survey of neighborhood-based recommendation methods," in *Recommender systems handbook*. Springer, 2011, pp. 107–144.

[162] R. Karimi, C. Freudenthaler, A. Nanopoulos, and L. Schmidt-Thieme, "Exploiting the characteristics of matrix factorization for active learning in rec-

ommender systems," in *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 2012, pp. 317–320.

[163] L. Zhao, S. J. Pan, E. W. Xiang, E. Zhong, Z. Lu, and Q. Yang, "Active transfer learning for cross-system recommendation." in *AAAI*. Citeseer, 2013.

[164] Z. Zhang, X. Jin, L. Li, G. Ding, and Q. Yang, "Multi-domain active learning for recommendation." in *AAAI*, 2016, pp. 2358–2364.