박사학위논문

추천 시스템 개선을 위한 무관심 아이템, 신뢰 네트워크, 카테고리 전문가 활용 방안

Improving Recommendation Systems by Exploiting Notions of Uninteresting Items, Trust Networks, and Category Experts

황 원 석

한양대학교 대학원

2016 년 2 월

박사학위논문

추천 시스템 개선을 위한 무관심 아이템, 신뢰 네트워크, 카테고리 전문가 활용 방안

Improving Recommendation Systems by Exploiting Notions of Uninteresting Items, Trust Networks, and Category Experts

지도교수 김 상 욱

이 논문을 공학 박사학위 논문으로 제출합니다.

2016 년 2 월

한양대학교 대학원

전자컴퓨터통신공학과

황 원 석

이 논문을 황원석의 박사학위 논문으로 인준함

2016 년 2 월

심 사 위 원 장 : 차 재 혁 (인)

심 사 위 원 : 김 상 욱 (인)

심 사 위 원 : 박 희 진 (인)

심 사 위 원 : 강 수 용 (인)

심 사 위 원 : 이 동 원 (인)

한양대학교 대학원

Table of Contents

List of Figures

List of Tables

국문 요지

추천 시스템 (recommendation system)은 유저가 가장 선호할만한 아이템을 자동으로 제공하는 기술으로, 그 중에서 협업 필터링 (collaborative filtering)은 가장 널리 연구되고 있는 방법이다. 협업 필터링은 유저들이 남긴 평점을 이용하여, 추천 대상인 액티브 유저와 취향이 유사한 이웃 유저들을 찾고, 이웃들이 선호하는 아이템을 추천한다. 그러나 대부분의 유저들은 소수의 아이템만을 평가하기 때문에 협업 필터링에서는 적은 수의 평점만을 이용할 수 밖에 없으며, 이로 인하여 정확도가 낮아지는 데이터 희소성 문제 (data sparsity problem)가 발생한다. 또한, 지속적으로 유저와 아이템이 증가하기 때문에 성능상의 문제가 발생한다.

본 학위 논문에서는 협업 필터링의 정확도 및 성능을 향상 시키기 위하여 (1) 무관심 아이템 (uninteresting item), (2) 신뢰 네트워크 (trust network), 그리고 (3) 카테고리 전문가 (category expert)를 이용하는 네 가지 방안을 제안한다. 무관심 아이템은 유저가 별다른 매력을 느끼지 못하여 이용조차 하지 않고 무시한 아이템이다. 유저는 이 아이템들에게 직접적으로 평점을 남기지 않았기 때문에 기존의 협업 필터링은 무관심 아이템들을 활용하지 못하였다. 우리는 협업 필터링에서 무관심 아이템을 고려하도록 함으로써 더 정확한 추천 결과를 도출하도록 한다. 또한, 무관심 아이템을 추천할 아이템 후보에서 완전히 제외하여 분석할 아이템의 수를 감소시킴으로써 성능 또한 향상한다.

신뢰 네트워크는 일종의 사회 연결망으로써 유저간의 신뢰 관계를 나타내는 네트워크이다. 기존의 협업 필터링에서는 액티브 유저의 이웃 유저를 찾는 대신 신뢰 네트워크에서 타겟 유저와 연결된 유저들을 활용하였다. 이 때, 액티브 유저가 신뢰하는 유저들 (trustees)만이 타겟 유저와 유사하다고 간주하였다. 우리는 액티브 유저를 신뢰하는 유저들 (trustors) 또한 액티브 유저와 유사한 취향을 가지고 있다고 가정한다. 이 가정을 바탕으로 Trustees 뿐만 아니라 trustors도 함께 이용함으로써 더 정확한 추천 결과를 도출할 수 있다.

이 아이디어를 확장하여, 본 논문은 신뢰 네트워크를 활용하여 데이터 대치 (data imputation) 방안을 제안한다. 기존의 데이터 대치 방법은 평점만을 이용하였으나,

평점과 상이한 신뢰 네트워크를 활용하면 더 정확한 대치 결과를 도출할 수 있을 것이다. 또한, 기존 대치 방법들과 달리, 정확하게 채울 수 있는 값만을 선별하여 채운다. 그 결과, 유저가 부여한 평점과 함께 대치 방법으로 추가된 평점을 이용하여 협업 필터링의 정확도를 향상할 수 있다.

협업 필터링에서 성능상의 문제를 야기하는 병목 중 하나는 액티브 유저의 이웃을 찾아내는 과정이다. 이를 해결하기 위하여 각 유저의 이웃을 찾는 대신 카테고리 전문가를 카테고리 별로 선별하고, 이들을 통해 추천하는 방안을 새롭게 제안한다. 우리는 카테고리 전문가를 해당 카테고리의 아이템을 가장 많이 평가한 유저로 정의한다. 그 결과, 카테고리 전문가를 선출 및 유지에 드는 계산 비용이 크게 감소하여 높은 성능을 유지할 수 있다. 또한, 실생활에서 유저는 전문가의 의견을 따르는 경향이 있기 때문에 정확도 측면에서도 손실이 없을 것이다.

우리는 각 접근 방안을 실제 데이터를 이용하여 정확도 및 성능을 평가하였다. 그 결과, 첫 번째 방안은 기존 방안에 비하여 약 5배 높은 정확도를 보였으며, 1.2 - 2.3배 짧은 시간에 추천 결과를 도출하였다. 두 번째 방안은 기존 방안보다 최대 1.9배 높은 coverage를 보였으며, 정확도 또한 2% 향상되었다. 세 번째 방안은 기존 방안보다 정확도가 약 3% 향상되었으며, 특히 추천이 어려운 cold-start 유저에 대한 추천의 정확도가 6% 향상되었다. 마지막으로 네 번째 방안은 기존 방안보다 5%의 정확도 향상이 있었으며, 9배 더 빠르게 수행되었을 뿐만 아니라, coverage 또한 10% 향상되었다.

1. Introduction

1.1 Background and Issues

Recently, a large number of people use the Internet to purchase various kinds of items such as movies, music, books, and groceries or to browse information such as reviews, news articles, and scientific articles. As the number of these items increases, it becomes more difficult for users to find their preferred items and information. For such users' convenience, *recommendation systems technology* is required [Ado05], which is a personalized service to provide an active user with a few items that she would like based on her history of evaluation, purchase, and browsing.

Figure 1.1 depicts how a recommendation system proposes items to an active user. In this example, the system recognized that a user is interested in shoes because she has purchased several pairs of shoes in the past. Based on this knowledge, it finally recommends new pairs of shoes that she has not known and is likely to be interested in. If the user buys some pairs of recommended shoes, the system reflects the purchases as well in the next recommendation.

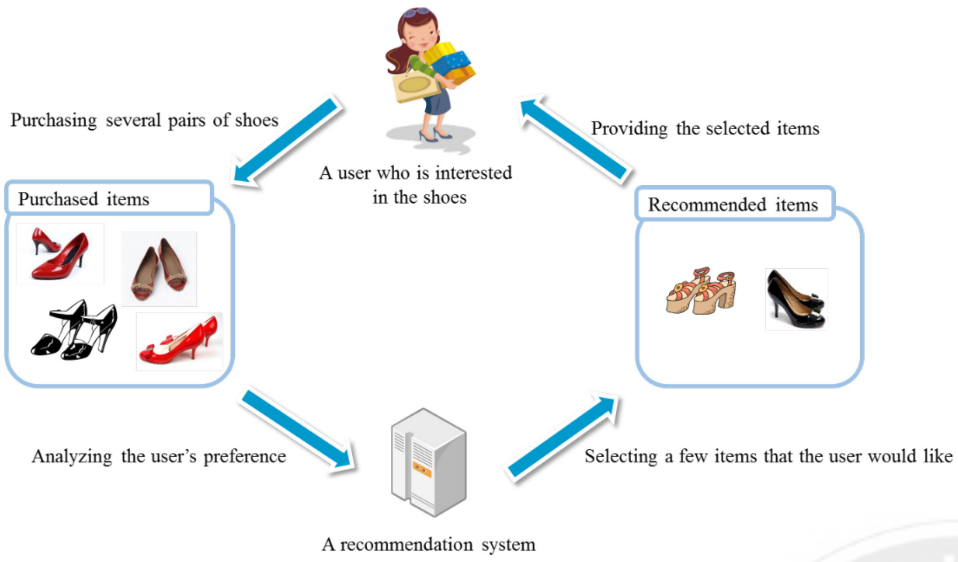


Figure 1.1. A process in the recommendation system.

Existing recommendation systems can be classified into three approaches: content-based [Som01, Zha01, Zha02], collaborative filtering [Lin03, Moh09, Sar01], and hybrid approaches [Sch02, Mid04, Pop01]. The content-based approaches recommend items that have contents similar

to those items that an active user has given high ratings. The collaborative filtering (CF) approaches recommend items that neighbors (who have similar taste to an active user) have given high ratings. The hybrid approaches produce a recommendation result by integrating the results obtained by both content-based and collaborative filtering approaches.

CF approach, one of the most popular and effective techniques in recommender systems, has been extensively studied in recent years. By and large, the CF approach can be applied to address two settings: (1) rating prediction and (2) top-*N* recommendation. The CF approaches aiming at the rating prediction focus on computing the absolute values of ratings that individual users would give to the yet unseen items. Other approaches considering the top-*N* recommendation find *N* items that individual users would most prefer. This dissertation focuses on both of two goals.

CF approaches can be evaluated in three aspects: (1) accuracy, (2) running time, and (3) coverage [Ado05]. First of all, CF approaches should be able to maximize users' satisfaction by recommending the preferable top-*N* items to them *accurately*. Second, they should be able to make a recommendation *in a reasonable time*. Third, CF approaches should be able to predict users' preference on items as many as possible. A low coverage indicates that the system may not recommend the most preferable items to an active user.

Meanwhile, there are some problems to achieve high accuracy, short running time, and high coverage in CF approaches. Most CF approaches focus on the users' *ratings*, which represent their preferences on items. However, most users evaluate only a few items, leading to the *data sparsity problem* [Bil98, Sar00, Hua04]. Specifically, the data sparsity problem is occurred when there are a lot of empty entries in a rating matrix *R* whose entry r_{ui} indicates user *u*'s rating on item *i*. Due to the lack of information, the CF approaches could suffer from *low accuracy and coverage* in this case. In addition, as the numbers of users, items, and ratings increase, *the times for analysis and prediction increase* in CF approaches.

To achieve three aspects of CF approaches, while addressing the data sparsity and long running time problems, this dissertation considers two strategies: (1) *densifying the rating matrix* and (2) *changing the neighbors*. The first strategy augments an original rating matrix (i.e., called a *densified rating matrix*) by filling values into the empty cells. We expect that the densified rating matrix can improve the accuracy, coverage, and running time.

Based on the densified rating matrix, CF approaches are able to analyze users' preference by utilizing more ratings, and thus, it is more accurate to predict ratings (or relative preferences). In addition, CF approaches improve the coverage because there are less unevaluated items whose

ratings cannot be predicted in the densified rating matrix. Furthermore, if CF approaches ignore some unrated items whose assigned rating values are low from recommendation candidates, the running time would improve.

Second strategy also improves three aspects by changing the neighbors of each active user. The more similar tastes to an active user the selected neighbors have, the more *accurate* ratings CF approaches can predict. In addition, if CF approaches select neighbors who evaluated more items (*i.e.*, have more ratings), their coverage can be improved because the neighbors provide more ratings for each pair of user and item. Moreover, if CF approaches select neighbors in a more simple way, their running time will decreases. This is because neighborhood selection is one of the bottleneck of many conventional CF approaches [Bre98, Sar01].

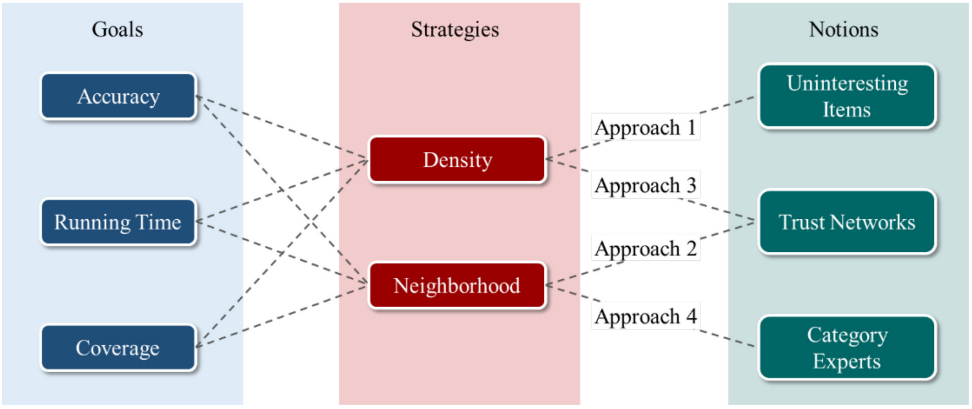


Figure 1.2. Goals of CF approaches and our strategies and notions.

Based on two strategies, this dissertation proposes four approaches by exploiting three notions: (1) uninteresting items, (2) trust networks, and (3) category experts as depicted in Figure 1.2. Based on first strategy, our first approach utilizes a concept of uninteresting items, *i.e.*, those items that are not attractive to users and unlikely to be purchased by users. The uninteresting items can be used as additional information in CF approaches to address the problems effectively by filling values into some empty cells corresponding with the uninteresting items. Similarly, a trust network (*i.e.,* a kind of social networks) is also taken as additional information in CF. Unlike existing CF approaches, which mostly focus on the trustees who are trusted by a certain user, our second approach also exploits the trustors who trust that user. In other words, our approach selects the neighbors different to the existing CF approaches based on the concepts of trustees and trustors. Furthermore, by extending this idea, our third approach imputes a rating matrix by exploiting a trust network to enrich the collaborative information. This approach improves the CF approaches by densifying the rating

matrix. Lastly, our fourth approach changes the neighborhood by utilizing the concept of category experts who know a lot of items in a certain category. Using the category experts, instead of neighbors, is more efficient in terms of running time because maintaining category experts is computationally more efficient than finding (or maintaining) neighbors.

1.2. Exploiting Uninteresting items

In this dissertation, as such, we strive to improve CF techniques with respect to its *accuracy* and *running time*. Our proposal is *method-agnostic* so that it can be applied to a wide variety of CF techniques seamlessly, improving their results universally. Our proposal is based on the following hypothesis in CF:

Hypothesis 1. Filling values into empty cells (*i.e.*, unrated items) in a rating matrix can improve the accuracy and running time of the top-*N* recommendation by CF methods.

Below, first, we develop several key ideas that are crucial to develop a solution based on the hypothesis.

Idea 1 (Exploit uncharted unrated items) Existing CF methods in general attempt to exploit the ratings that users have provided, yet the fraction of known ratings in a rating matrix is in general quite small. Suppose, for a rating matrix R with m users and n items, on average, a user rates k items. Then, the fraction of rated items in R is: $\frac{m \times k}{m \times n} = \frac{k}{n}$. As it is often common for an e-business to sell millions of items with a very long tail and a user on average rates only a small number of items, asymptotically, such a fraction of rated items in R becomes extremely small (*i.e.*, $k \ll n$). As such, we believe that exploiting the vast number of uncharted "unrated" items in R can lead to a significant improvement in CF.

Several existing works assume that items are not evaluated (*i.e.*, unrated) when a user is not interested in them. Nan Hu et al. [Hu09] and Nilesh Dalvi et al. [Dal13] pointed out that most rated items have either 4 or 5 rating scores out of [1 ... 5], 5 being the best. In other words, users tend to evaluate only those items that they liked (thus high ratings among rated items), while ignore items that they dislike in advance. Similar to this finding, Harald Steck [Ste10] presumed that unrated items are less likely to be preferred by users. However, we emphasize that users are able to know only some of unrated items, not all of them.

Idea 2 (Some unrated items are uninteresting) We observe that unrated items in R can be categorized into several types: (1) unrated items that users were not aware of their existence, (2)

unrated items that users knew and purchased but did not rate, and (3) unrated items that users knew but did not like and thus did not purchase. In particular, we note that the unrated items of the third type, named as uninteresting items (I^{un}), clearly indicate users' latent "negative" preference. Therefore, we believe that it is better not to recommend those uninteresting items. Challenge is however that it is not easy to identify I^{un} in R.

Idea 3 (Uninteresting items have low pre-use preferences) In order to identify I^{un} in R, we first have to understand a user's pre-use preference to items–*i.e.*, a user's impression on items before purchasing and using them. Then, based on this novel notion, we can rephrase I^{un} as those items whose pre-use preferences are not high. Unfortunately, ratings that users leave in R do not indicate pre-use preferences but the preferences "after" using the items, named as post-use preferences. However, also note that users must have had high pre-use preferences to all rated items (otherwise, users would have not purchased them in the first place). Therefore, we believe that we can utilize the pre-use preferences of rated items to infer the latent pre-use preferences of unrated items, and finally identify I^{un} with low pre-use preferences.

With I^{un} in R found, next, we propose to improve top-N recommendation via two strategies: (1) excluding I^{un} from the top-N recommendation results, and (2) exploiting both I^{un} and ratings to predict the relative preferences of items better. The first strategy is likely to improve top-N recommendation. As I^{un} with low pre-use preferences are by definition the items that users were aware of their existence but did not like in the first place, they are likely to be false positives if included in top-N recommendation. Therefore, proactively, we exclude I^{un} from the top-N recommendation results.

Our second strategy is also likely to improve top-N recommendation by recommending preferred items more accurately. We can explain the effect of second strategy through the concept of typical memory-based CF methods that depend on the fact that a user u is more likely to prefer items that her neighbors also like. Suppose a few neighbors rated an item i highly but most neighbors considered i as an uninteresting item (thus left i unrated in R). In this situation, existing CF methods are likely to recommend i to u as her a few neighbors rated them high. On the other hand, if we consider the fact that many more neighbors actually viewed i as *uninteresting*, we could avoid recommending i to u.

Based on the three ideas and two strategies, now, we propose a novel solution to apply the notion of *uninteresting items* to the framework of any existing CF methods. To the best of our knowledge, ours is the first work to exploit the notion of uninteresting items in improving CF methods. Our

proposed solution consists of three steps: (1) it infers the pre-use preferences of unrated items by solving the *one-class collaborative filtering* (*OCCF*) problem [Pan08], [Sin09], (2) it assigns zero ratings to the found uninteresting items in R whose pre-use preferences are not high, yielding an augmented matrix Z, and (3) it applies any of existing CF methods to Z, instead of R. This simple-yet-novel imputation not only addresses the sparsity problem by enriching a rating matrix but also completely prevents uninteresting items from being recommended as top-N items, thereby improving the overall accuracy greatly.

1.3. Exploiting Trustors in a Trust network

In CF approaches, the *data sparsity problem* results in low accuracy and coverage. In particular, CF techniques tend to perform poor for users who gave ratings to only a few items, so called cold-start users, as it is difficult to infer those users' latent interests. As one way to remedy these problems, *trust-aware approaches* to collaborative filtering [Gol05, Mas07, Jam09] have been proposed, which utilize the so-called a *trust network*. A *trust network* is a kind of a social network, which represents trust relationships among users. A trust network establishes a trust relationship between two nodes [Han14]. For instance, in the trust network of Figure 1.3, the nodes and edges represent users and trust relationships, respectively, e.g., the edge ($U_1 \rightarrow U_4$) represents "U_1 trusts U_4." In this case, we call U_1 a trustor and U_4 a trustee. Note that a user can be a trustor and a trustee at the same time. For example, U_4 is a trustee of U_1, and a trustor of U_5.

Trust-based recommendation methods [Gol05, Mas07, Jam09] predict a rating on a target item for an active user by exploiting the ratings on that particular item given by so-called "trustable" users, instead of "similar" users in the conventional collaborative filtering approaches. In these methods, trustable users are defined to be those users in the trust network reachable within certain distance from the active user by following the links of trust relationships in the forward direction, i.e., from a trustor to a trustee. In Figure 1.3, for instance, trustable users of U_1 are $\{U_4, U_5, U_8\}$. That is, given an active user U_1, those users who U_1 trust directly are trustable (U_4 and U_8), those who these trustable users trust are also trustable (U_5, who U_4 trusts), and so on.

This trust-based approach to collaborative filtering has advantages over the conventional approaches in that it can recommend items even for cold-start users effectively. This is because, even for a user who gave almost no ratings, a large group of *trustable* users can be easily formed by traversing through the trust network in this approach, whereas it would be extremely difficult to

identify similar users by analyzing rating patterns if we follow the conventional approach. In a trust network, even a cold-start user may have direct links to many other users, or she may be linked to some users who have links to many other users [Jam09].

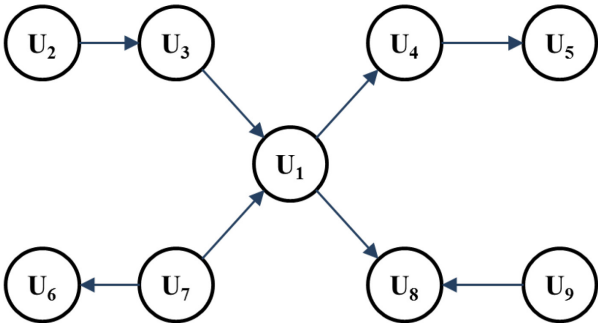


Figure 1.3. Example of a trust network.

The rationale behind utilizing ratings information of trustable users instead of similar users is the assumption that a person may well be interested in those items which her trustees are interested in because one tends to not trust people with entirely different tastes [Mas04]. This assumption has been validated through experiments by existing trust-based methods which showed that predicted ratings based on trustable users perform fairly well [Gol05, Mas07, Jam09, Ma08, Jam10, Ma10].

In this dissertation, we extend this assumption such that a person can also be interested in those items which her trustors are interested in, that is, we regard *the trust relationship as symmetric*. However, most existing methods have overlooked this possibility. This dissertation aims to exploit this addition for improving ratings prediction, especially in light of the coverage. As a result, the definition of trustable users is now to include all those users reachable from the active user through the trust relationship *regardless of the link directions*. We envisage that this approach will induce a larger group of trustable users, improving the coverage[1] but still maintaining the accuracy when compared to existing methods which use forward links only. Especially, when predicting the ratings for cold-start users, using backward links as well as forward links may generally improve the coverage, because the number of trustable users becomes larger in most cases.

In collaborative filtering, however, utilizing a larger group of similar users is likely to result in decreased accuracy due to possible inclusion of dissimilar users in the set. In our approach, nevertheless, accuracy will not be compromised because both trustors and trustees are assumed to

[1] Although the existing trust-based approaches are useful, but there is some room for improving the coverage.

have similar interests to the active user. In this dissertation, we will verify this assumption by experimenting with the *trustors group* formed by following the backward links and show that the accuracy does not suffer. We will also analyze the datasets to show that using both forward and backward links improves the coverage as compared to the case of using forward links only.

For the purpose of comparative analysis, we define three types of trustable user sets reachable from an active user u through the trust links in a trust network: $FWD(u)$, $BWD(u)$, and $UND(u)$. $FWD(u)$ represents the set of users reachable from u through the forward links, $BWD(u)$ the set of users reachable from u through the backward links, and $UND(u)$ the set of users reachable from u through the links regardless of the directions (i.e., obtainable from the corresponding undirected graph).

Note that in fact $FWD(u)$ refers to the set used by existing trust-based recommendation methods [Gol05, Mas07, Jam09, Mas05]. These methods impose a *distance limitation* in the length of the linked chains because users who are far away from the active user can have different interests, or are noisy and unreliable. Similarly, we will impose a distance limitation for all these three sets.

DEFINITION 1.1 (TRUSTABLE USER SETS). For a user u, when the distance limitation is set as δ, trustable user sets, *i.e.*, $FWD(u)$, $BWD(u)$, and $UND(u)$, are defined as: $FWD(u) = \{v | d^f(u,v) \leq \delta\}$, $BWD(u) = \{v | d^b(u,v) \leq \delta\}$, and $UND(u) = \{v | d^u(u,v) \leq \delta\}$ where $d^f(u,v)$, $d^b(u,v)$, and $d^u(u,v)$ indicate the distance between two nodes u and v with regard to forward links, backward links, and both of forward and backward links, respectively.

In the trust network of Figure 1.3, for example, $FWD(U_1)=\{U_4, U_5, U_8\}$ as shown earlier, $BWD(U_1)=\{U_2, U_3, U_7\}$, and $UND(U_1)=\{U_2, U_3, U_4, U_5, U_6, U_7, U_8, U_9\}$ if no distance limitation is imposed. Specifically, $UND(U_1)$ includes U_9 because U_8 is connected through a forward link and U_9 is connected to U_8 through a backward link. In this dissertation, we will investigate the effects of utilizing these three sets by comparing their results of ratings prediction.

In this dissertation, the following questions will be addressed through experimentation using a real-world trust network.

- Do the trustable users in the three sets, $FWD(u)$, $BWD(u)$, and $UND(u)$, have actually similar interests to user u? Will these three sets lead to similar or different results of recommendation?
- Will we achieve increased coverage by utilizing $UND(u)$? Will this case result in similar accuracy to the case with $FWD(u)$ only?

- Will we achieve increased coverage in predicting ratings for cold-start users? Will this case result in similar accuracy to the case with $FWD(u)$ only?

1.4. Trust-Aware Imputation

An imputation approach is one of effective approaches that deal with problems of the data sparsity and cold start users. The basic idea of this approach is to assign proper values to the missing ratings implying users' unevaluated items. In this approach, estimation of assigned values is the most important part that affects accuracy of CF approaches. Specifically, as an assigned value is more similar to a rating that a user would give, CF approaches show better accuracy. For estimating the assigned values, the existing methods only consider the users given ratings although there is additional helpful information, e.g., a trust network. To improve the accuracy of estimating assigned values, this dissertation proposes a novel imputation method, called *trust-aware imputation method*, which considers a trust network for computation of assigned values.

Our method estimates the unrated ratings in a rating matrix and then applies the properly *densified* rating matrix and applies it to a recommendation procedure that employs probabilistic matrix factorization [Kor09]. The proposed imputation method builds upon the probabilistic matrix factorization procedure which performs well on large and imbalanced datasets requiring less learning time than other matrix factorization approaches [Kor09].

Specifically the proposed trust-based imputation method estimates the rating of an item for a user as aggregating the ratings given by those users, called *reliable neighbors*, who have similar tastes to her, and finds similar users by examining a trust network. The *trust network* is a kind of a social network representing trust relationships among people. Noticeably a user has similar preferences to those users who are trusted by that user [Gol05, Jam09, Jam10, Mas07, Ma08, Ma09b], so we can find reliable neighbors of each user through the trust network. In this dissertation, we utilize only trust networks as a source of information on relationship, but the proposed imputation method is easily applicable to social networks other than trust networks.

Indeed the accuracy of the proposed imputation method is affected by the *number* of selected reliable neighbors and *similarity* between a target user and her reliable neighbors. Specifically, the more and similar users are selected as reliable neighbors, the more accurate results are produced. To maximize this condition, in our method, the reliable neighbors of a certain user represent those users who are reachable from her through the bi-directional connection of edges in the trust network. In

this way, we can include the more users compared to the neighbors defined in existing methods [Gol05, Mas07]. This is because the existing methods focus on only those users (trustees) who are trusted by a certain user. Furthermore, our reliable neighbors have similar tastes to a target user because the existing researches [Ma08, Jam09] show that a user has similar tastes to trustees, and the similarity relationship is inherently symmetric.

In addition, the reason why we also select those users who are not directly connected but reachable to a user in the trust network as her reliable neighbors is that those users possibly bear a similarity to her. However, the greater the distance between two users in a trust network, the more different their preferences. Based on this assumption, in setting trust reliable neighbors of a user we select those users within a certain distance level from the user. The adopted approach reasonably excludes the users who are likely to have different preferences from the reliable neighbors.

In order to enhance the accuracy, our method imputes only a part of missing ratings rather than whole missing ratings. For a user, there are many missing ratings whose corresponding items are evaluated by few reliable neighbors. In this case, it is hard to estimate the values for those missing ratings accurately because the estimation is easily biased by few reliable neighbors. For this reason, our method omits to impute some missing rating whose estimated values would be incorrect.

Previous recommendation methods, called the trust-based recommendation methods [Gol05, Mas07, Jam09], already use trust networks for alleviating the data sparsity problem. All of these methods predict the ratings of an active user by looking up ratings of those users who are reachable from the user through only forward directions of edges in the trust network. The previous methods ignore users who are reachable through backward directions of edges, so their rating predictions are inevitably based on a fewer users. Unlike those methods, our imputation method explores more users who have similar preference; consequently, unrated ratings can be more accurately estimated.

Unlike our method, the previous methods are at a risk of including users who have different tastes to an active user, possibly causing the accuracy to decrease. For example, several existing trust-based recommendation methods [Ma08, Ma09b, Jam10] utilize users whose tastes are different from that of an active user because the users far from an active user can affect estimation of ratings. In addition, although the proposed method is similar to the previous imputation approaches for recommendation [Bri98, Ma07, Ren12], to the best of our knowledge, there has been no recommendation method that employs a matrix factorization based imputation approach with bi-directional trust structures.

1.5. Using Category Experts

The previous recommendation systems are often based on neighborhood-based methods (NBM) [Su09], which predict a rating of an item that a user has not evaluated yet and suggest her preferable items based on the predicted ratings. NBM has several advantages such as simplicity, justifiability, efficiency, stability, and clear reasoning behind recommendation [Rob07, Sha11]. It finds the neighbors who rate items in a close range to the active user for ratings prediction using the ratings of those neighbors [Res94]. Some NBMs search for neighbors at every recommendation request to include the latest ratings. This approach can find more similar users because they use more ratings for finding the neighbors, but suffers a high execution time in searching. On the other hand, another group of NBMs pre-computes similarities or pre-builds the models by ignoring the latest ratings to reduce the execution time. However, those NBMs would have an accuracy problem because they ignore a portion of ratings (i.e., the latest ratings). Thus, NBMs have a *tradeoff between running time and accuracy*.

To address this challenge, we propose new recommendation methods based on 'category experts' (referred to as CE) rather than neighbors of NBM. In a real world, users have a tendency to trust expert opinion, so preference prediction with the experts would be accurate [Ama09, Cho07]. We define the category experts as top-k users in giving ratings of a certain category. k is determined by the empirical analysis on each category rating. CE aggregates the ratings given by the experts to predict ratings that the user will give to unevaluated items. We also extend CE by exploiting two metrics to improve the accuracy: 'similarity' between users and category experts and 'category interest,' defining the degree of interest of the users.

CE improves the running time significantly because the experts can be identified and maintained easily by counting the number of ratings given by each user. In addition, its accuracy is not worse than the existing methods since people want to look for advice from experts of specific fields in a real world. Thus, using the category experts leverages the tradeoff between running time and accuracy. CE and its extensions improve the coverage, a portion of unseen items that are actually predictable because category experts tend to have rated more items than similar users.

1.6. Organization

The dissertation is organized as follows. Section 2 addresses the related work. Section 2.1

addresses the conventional recommendation systems including collaborative filtering. In Section 2.2, we explain existing work related to the unrated items. Section 2.3 deals with existing CF approaches using the trust network. Section 2.4 addresses the imputation methods for recommendation. Section 2.5 presents the CF approaches exploiting the experts.

Section 3 deals with the uninteresting items in CF. In Section 3.1, we explain the preliminaries of our approach. In Section 3.2, we present our approach in detail. In Section 3.3, we evaluate our approach in comparison with existing ones via extensive experiments.

Section 4 shows an approach for exploiting trustors as well as trustees in the trust network for recommendation. In Section 4.1, we explain existing trust-aware CF methods to be applied our approach. Section 4.2 presents our approach by defining the three sets of trustable users in a trust network, and analyzing the similarity of these sets. Section 4.3 shows the experimental results of ratings predicted by applying our approach to existing trust-based recommendation methods.

Section 5 deals with an imputation method using trust network. We introduce our proposed model in section 5.1. Our experiments are reported in section 5.2. Section 6 deals with the CF approach employing the category experts. Section 6.1 proposes the CE method and its extensions. Section 6.2 deals with the running time and accuracy issues with the real-world datasets. In Section 7, we conclude the dissertation and discuss the limitation of proposed approaches and future work.

2. Related Work

2.1. Conventional Recommendation Systems

Most existing recommendation methods utilize ratings provided by users. Typical approaches based on ratings are content-based, collaborative filtering, and hybrid methods. The content-based approach such as TF-IDF and Bayesian classifier, recommends the items with the features similar to the ones the active user preferred in the past [Som01][Zha01][Zha02]. It is difficult to recommend items to a new user since she does not have rated a sufficient number of items. In addition, the content-based approach cannot recommend items that are different from anything the user has seen before, In other words, it recommends the same type of items that a user purchased. So, this recommendation causes that the users lose their preference in recommended items. The content-based approach recommends multimedia data (*i.e.*, audio and video streams) because it is difficult to extract features form them.

The collaborative filtering approach recommends items that the people with similar tastes and preferences liked in the past. This approach can be further classified into user-based [Shi09, Bre98, Han01, Jam09, Kon09, San07], item-based [Lin03, Moh09, Sar01], clustering-based [Ung98, Sar02], and latent-based methods [Kor09, Sal08, Sal07b, Ren05]. The user-based method looks for k users who have a similar preference with the active user and predicts the rating of an active user on a target item from the ratings given by k users. Its similarity can be calculated via Pearson's correlation coefficient, cosine similarity, or the extended generalized vector-space model [Sob99]. The predicted rating $\hat{r}_{u,i}$ of item i for user u is computed by $\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{v \in S_u}(r_{v,i} - \bar{r}_v) \times |sim(u,v)|}{\sum_{v \in S_u}|sim(u,v)|}$ where $r_{v,i}$ is the rating by user v on item i and $\bar{r}_u$ is the average of all the ratings assigned by an active user u. $sim(u, v)$ is the similarity between user u and v, and S_u is the set of k users (neighbors) that are the most similar to active user u. The similarities between the active user and others are calculated online upon a recommendation request. Herlocker et al. [Her99] analyzes the accuracy of UBM with respect to the selected neighbors having similar tastes. To identify a closer neighborhood, the following approaches also have been proposed: Default voting, inverse user frequency, case amplification [Bre98], and weighted-majority prediction [Del99, Nak98]. UBM can provide accurate recommendations because it considers all ratings including the latest ratings for searching neighbors. The latest ratings are important because most users assign ratings to only a few items

[Hua04, Bil98, Sar00].

The item-based methods were proposed to overcome a running time problem by pre-calculating similarities among all items [Kar01, Sar01]. It calculates the similarity of items rather than that of users because the relationships between items are relatively static [Sar01]. It predicts the active user's rating on a target item by referencing her ratings on those items similar to the target item. The predicted rating $\hat{r}_{u,i}$ assigned by active user u on target item i can be calculated as $\hat{r}_{u,i} = \frac{\sum_{j \in S_i}(r_{u,j} \times |sim(i,j)|)}{\sum_{j \in S_i}|sim(i,j)|}$ where $sim(i,j)$ represents the similarity between user i and j, and S_i is the set of items that are similar to item i.

The clustering-based methods [Ung98, Sar02] improve the running time by determining the neighbors of each user before the recommendation request. Several clustering-based method variations use various clustering models [Est96, Ank99, Han01] to classify the users off-line. They make user clusters before the recommendation request, and then predict the active user's rating of item i based on the ratings given by those users who belong to the group (i.e., cluster) that the active user belongs to. In addition, there are other model-based methods that use the Bayesian model [Miy00] and the latent semantic model [Hof04] other than clustering models.

The latent-based methods [Kor09, Sal08, Sal07b, Ren05] all focus on fitting the rating matrix using low-rank approximations making further predictions. The premise behind a low-rank factor model is that there are only a small number of factors influencing preferences and that a user's preference vector is determined by how each factor applies to that user.

The hybrid approach [Sch02, Mid04, Pop01] combines content-based and collaborative filtering approaches. There are various combination techniques for the hybrid approach such as voting, linear combination, and selecting one of the two approaches depending on the situation. It has the problem about combining the recommendations from two approaches and is plagues with the new user problem similar to that of content-based and collaborative filtering approaches.

2.2. Work Related to the Uninteresting Items

Most CF methods, despite their wide adoption in practice, suffer from low accuracy if most users rate only a few items (thus producing a very sparse rating matrix), called the *data sparsity problem* [Hua04]. This is because the number of unrated items is significantly more than that of rated items. To address this problem, some existing work attempts to infer users' ratings on unrated items based

on additional information such as clicks [Liu10] and bookmarks [Niw06]. While reporting improvements, however, these works require an overhead of collecting extra data, which itself may have another data sparsity problem. Compared to [Liu10, Niw06], our proposal does not require any extra data and solely work using an existing rating matrix.

In addition, to improve the accuracy of recommendation, other works attempted to leverage both ratings and the fact whether a user evaluates an item or not. For instance, SVD++ [Bel07, Kor08] builds an extended SVD model exploiting both information. The conditional restricted Boltzmann machine (RBM) [Sal07a] and constrained probabilistic matrix factorization (PMF) [Sal07b] also account for both of information in learning their models. However, these approaches have a rather simplistic assumption such that a user would dislike all unrated items. On the other hand, we strive to discern a subset of unrated items that users truly dislike. As such, as demonstrated in experiments, our proposal yields several orders of magnitude improvements in accuracy, compared to these approaches (*e.g.*, SVD++).

Finally, several CF methods (*e.g.*, [Cre10, Mar09]) have proposed to fill missing ratings with a particular value in order to improve the accuracy. They also simply assume that a user would dislike all unrated items. Based on this assumption, for instance, PureSVD [Cre10] fills all missing ratings with zeros, and then makes prediction using both known ratings and zero ratings. Marlin et al. [Mar09] assign a low value to all missing ratings, and then make recommendation by learning a multinomial mixture model. By filling *all* missing ratings with low values, however, this approach could mistakenly assign low values to the items that users might like, thereby hurting an overall accuracy in recommendation. While the idea of "zero-injection" in our approach is analogous to these works, our approach selectively applies the zero-injection to only uninteresting items with low pre-use preferences, maximizing the impact of filling missing ratings.

2.3. Work Using Trust Networks

Trust-based recommendation methods, as another solution for the problems, have drawn considerable attention recently [Gol05, Mas07, Jam09, Ma08, Ma09a, Jam10]. TidalTrust [Gol05] performs a modified breadth-first search in a trust network to predict a user rating for recommendation. Massa and Avesani [Mas07] introduces MoleTrust, which has a similar idea with TidalTrust. However, when finding users who rated a target item, MoleTrust only considers users in the trust network up to a maximum-depth which is independent of any specific user and item.

TrustWalker [Jam09] supposes a random walker [Bri98] who starts from an active user (node) and moves to the other nodes through the edges on the trust network.

Some recent recommendation methods via matrix factorization use trust networks as well. Proposing a trust-based recommendation method [Ma08], Ma *et al.* also developed a method, called STE, which combines probabilistic matrix factorization and social networks [Ma09a]. Considering a rating matrix and a trust matrix, the method factorizes the rating matrix using latent user features and latent item features, and factorizes the trust matrix in the same manner. The method [Jam10] incorporates the users who are reachable from an active user to improve the quality of recommendation.

2.4. Imputation Methods

Some methods which estimate unrated ratings by imputation have been proposed. Default voting [Bre98] is a straight forward imputation-based method which assumes default values for those missing ratings, such as average ratings by a small group of users for other items in order to increase the size of a co-rated item set. Ma *et al.* developed a data imputation method [Ma07] which took account of users' confidence. They set a threshold to decide whether a user is a confident or not; and only fill the missing data to users that have highly confident reliable neighbors. Ren *et al.* proposed an auto-adaptive imputation method [Ren12] which aims to address the data sparsity problem. They identify a set of key ratings for a pair of an active user and an active item, and adaptively impute them according to ratings of each user. Those previous imputation-based recommendation methods show better accuracy compared to the previous collaborative filtering methods. However, most of those methods ignore additional information such as a trust network and take no advantage of matrix factorization which has proved to be quite effective in many researches [Kor09, Sal07b].

2.5. Work Using Expert

There are several previous methods considering experts to improve the accuracy. Amatrin et al. [Ama09] utilize the ratings given by experts in Rotten Tomatoes in order to recommend to Netflix users. Yun et al. [Yun01] also crawl the expert ratings from Rotten Tomatoes and predict the ratings for users in the other domain via the SVD-based model [Zha05]. However, these methods showed an additional overhead for crawling the expert ratings from other domains like Rotten Tomatoes. If

there are no pre-defined experts, they cannot provide recommendations.

Cho et al. [Cho07] define experts as the users who evaluate a lot of items. It predicts the rating on the target item by considering ratings from both experts and neighbors who have a similar taste to the active user. Thus, it suffers from long execution time of searching for experts and neighbors. Liu et al. [Liu11] introduce 'star' users, which are close to experts. The star users are virtual users who represent interests of whole users, and a training algorithm selects the star users through analyzing their ratings.

Pham et al. [Pha13] propose an expert-based method for interactive recommendation systems. This method defines experts for a given active user and her attributes (i.e., genre, actor, and director). To define the experts, it needs to analyze the attributes as well as correlation among users. Pham et al. [Pha14] also propose another expert-based method that corrects the users' ratings based on experts' opinions for more accurate recommendations. In order to find the experts, the method uses a <A,V>-SPEAR algorithm that refers to an ontology built based on items' attributes. To our knowledge, the existing expert-based recommendation methods ignore the running time issue. In this dissertation, we propose methods based on category experts instead of neighbors to balance the trade-off between running time and accuracy of NBMs.

3. Exploiting Uninteresting Items

3.1. Preliminaries

In this section, we introduce the details of uninteresting items by contrasting them to interesting items, rated items, and unrated items. Furthermore, we clarify the difference between pre-use preferences and post-use preferences.

A user u's pre-use preference to an item i is a different notion from a rating given to i in a rating matrix, which really represents u's post-use preference to i. Existing CF methods mainly attempt to exploit ratings (thus post-use preferences). Let us first introduce a few basic notations used throughout this chapter. Let $U = \{u_1, \cdots, u_m\}$ be a set of m users, $I = \{i_1, \cdots, i_n\}$ be a set of n items, and r_{ui} be the rating given to item i by user u. A corresponding rating matrix is referred to as $R = (r_{ui})_{m \times n}$, and p_{ui} (resp. q_{ui}) indicates user u's pre-use (resp. post-use) preference on item i. In theory, both types of preferences p_{ui} and q_{ui} exist on a user-item pair (u, i) although in reality they are not always computable or available. Table 3.1 summarizes key notations used in this chapter.

Table 3.1. Notations used in this chapter

Notation	Description
p_{ui}	User u's pre-use preference on item i
q_{ui}	User u's post-use preference on item i
r_{ui}	A rating given to item i by user u
P	A pre-use preference matrix whose entry is p_{ui}
Q	A pre-use preference matrix whose entry is q_{ui}
R	A rating matrix whose entry is r_{ui}
I_u^{un}	A set of uninteresting items for user u
I_u^{in}	A set of uninteresting items for user u
I_u^{pre}	A set of preferred items for user u

Note that u has pre-use preference on i based on its *external* features that u could obtain without actually using i (*e.g.*, genre or director information, in case of a movie). After using i, based on the level of her satisfaction, u then assigns a specific rating to i, indicating her post-use preference for i. The post-use preference is therefore determined by the *inherent* features that u had not known before using i (*e.g.*, storyline or choreography of a movie). Let us contrast two preference types in a more

detail using the following example.

EXAMPLE 1 (TWO PREFERENCE TYPES). Figure 3.1 illustrates the pre-use and post-use preferences of a user *u* for three movies. Initially, *u* has a high pre-use preference for Movie #1 and Movie #2. On the other hand, *u* does not have high pre-use preference for Movie #3. In that case, Movie #1 and Movie #2 are to be called as *interesting* items to u while Movie #3 as an *uninteresting* item. Thus, *u* would decide to watch only two movies with high pre-use preferences. After watching the movies, *u* likes Movie #1 as she expected but is disappointed at Movie #2. Therefore, *u* assigns a high rating to Movie #1 and a low rating to Movie #2. In contrast, *u* does not watch Movie #3 as it is an uninteresting movie to her. Her post-use preference to Movie #3 *remains unknown* (*i.e.*, empty in a rating matrix).

Movie	Pre-use preference	Post-use preference	Rating
Movie#1	High	High	5
Movie#2	High	Low	1
Movie#3	Not high	Unknown	Unrated

Figure 3.1. Pre-use, post-use preferences, and ratings for three movies.

Figure 3.2 further depicts how a user *u* thinks about an entire set I of items with the *Venn diagram*. Let *interesting items* I_u^{in} denote items with high pre-use preferences while *uninteresting items* I_u^{un} denote *u*'s items with not-high pre-use preferences. The two item sets are disjoint, *i.e.*, $I_u^{in} \cap I_u^{un} = \emptyset$, $I_u^{in} \cup I_u^{un} = I$. (In Section 3.2.2, we will explain how to identify uninteresting items from entire item set *I*.) We formally define them as follows:

DEFINITION 1 (UNINTERESTING ITEMS). For a user *u*, a set of uninteresting items I_u^{un} is defined as: (1) $I_u^{un} = I - I_u^{in}$ and (2) $p_{ui} \leq p_{uh}$ $(\forall i \in I_u^{un},\ \forall h \in I_u^{in})$.

Among interesting items I_u^{in}, user *u* buys/uses some items and evaluates them by assigning ratings. A set of items that are likely to get high ratings are called *preferred items*, denoted by I_u^{pre}, which is a subset of interesting items as shown in Figure 3.2 (*i.e.*, $I_u^{pre} \subseteq I_u^{in}$).

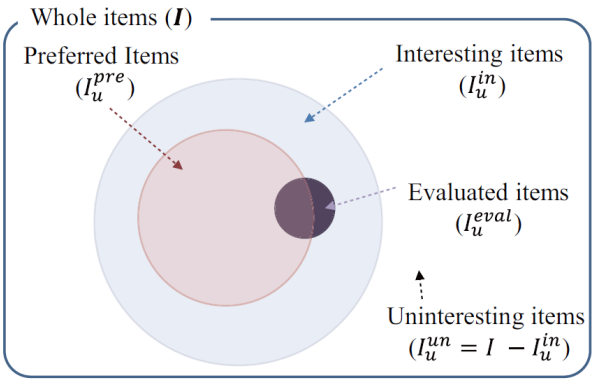


Figure 3.2. Venn diagram for the preferences and interestingness of items.

In a real scenario, u would be able to evaluate only a small fraction of interesting items. This item set, denoted by I_u^{eval}, is only a subset of I_u^{in} (*i.e.*, $I_u^{eval} \subseteq I_u^{in}$). For this reason, if we understand the uninteresting items of each user, we can understand the users' taste more accurately.

Based on this viewpoint, our goal is to identify top-N preferred items to user u by considering the uninteresting items for each user. Specifically, user u's pre-use and post-use preferences for item $I \in I_u^{eval}$ are known while both types of preferences for item $j \in I - I_u^{eval}$ are unknown. If u has evaluated item i, its pre-use preference p_{ui} can be considered high. Based on known pre-use preferences, we infer the pre-use preferences of the remaining unknown items j. In addition, its post-use preference q_{ui} can be directly indicated by the score of r_{ui}. We formally define our problem for top-N recommendation as follows:

PROBLEM 1 (TOP-N RECOMMENDATION) For user u, identify top-N items $J = \{j_1, \cdots, j_N\}$ such that: (1) $J \subseteq I_u^{in} - I_u^{eval}$ and (2) $q_{uj_1} \geq \cdots \geq q_{uj_N} \geq q_{uy}$ $(\forall y \in I - I_u^{eval} - J)$.

3.2 Proposed Approach

Existing CF methods employ user u's preferences only on evaluated (rated) items $i \in I_u^{eval}$. On the other hand, our approach identifies uninteresting items I_u^{un} and exploits them to enrich the rating matrix with additional user ratings. In addition, CF methods equipped with our approach could exclude those uninteresting items explicitly from top-N recommendation. We carefully analyze the benefits of our approach in Section 3.2.4.

The main challenges of our approach are two-fold: (1) *how to identify uninteresting items among unrated items* and (2) *how to exploit uninteresting items thus discerned in CF methods*. To address

the first challenge, we need to infer the pre-use preferences for all of the unrated items, and to find those unrated items whose pre-use preferences are not high. For the second challenge, we build the zero-injected matrix whose some entries are set as 0 if their corresponding items are considered *uninteresting*. This augmented matrix can be applicable to *any* CF methods (thus making our approach method-agnostic), which enables those CF methods to benefit from uninteresting items in their predictions.

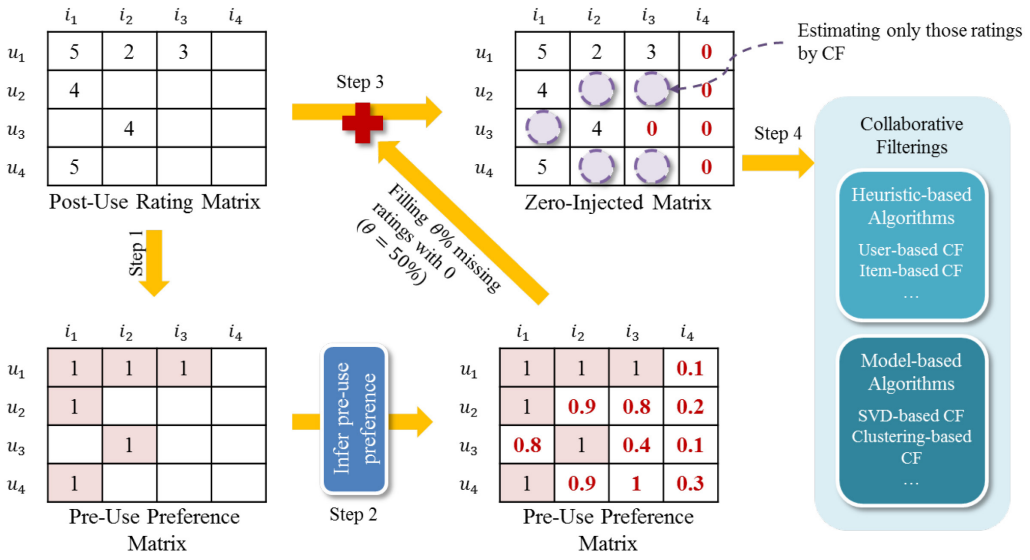


Figure 3.3. Overview of our approach.

Figure 3.3 depicts the overall process of our proposed approach. First, we build a *pre-use preference matrix* $P = (p_{ui})_{m\times n}$ by examining a rating matrix $R = (r_{ui})_{m\times n}$. Specifically, we set the pre-use preference, p_{ui}, of a user u on an item i as 1 when $r_{ui} \in R$ has been already rated (*i.e.*, u must have liked i so that she bought i). (Step 1). The value 1 is the highest pre-use preference because we represent p_{ui} as a real value in [0, 1]. Next, we infer (unknown) pre-use preferences on "unrated" user-item pairs (u, i) (*i.e.*, p_{ui} = null) based on other observed pre-use preferences (*i.e.*, $p_{ui} = 1$) and add them in P, which becomes $\hat{P}$ (Step 2). We define uninteresting items for each user based on $\hat{P}$, and build a *zero-injected matrix* $Z = (z_{ui})_{m\times n}$ (Step 3). That is, z_{ui} is set as r_{ui} when user u has actually evaluated item i and is set as 0 if i is an uninteresting item for u. In our approach, an item j is an uninteresting item for a user v if j's pre-use preference score $\hat{p}_{ui}$ is ranked within the bottom θ% in $\hat{P}$. Thus, Z is an augmented matrix that contains u's original ratings as well as "0", inferred and injected by our solution. We then apply a CF method to Z to predict the

post-use preferences of empty entries (dotted circles) in Z (Step 4). Finally, we recommend top-N items to an active user. In the following subsections, we explain each step in detail.

3.2.1 Inferring Pre-Use Preferences

It is straightforward to determine a pre-use preference p_{ui} when a user u has already rated an item i (*i.e.*, $r_{ui} \neq null$) because i must be interesting to u in the beginning, *i.e.*, $I_u^{eval} \subseteq I_u^{in}$. As such, we set the pre-use preference p_{ui} as 1 in this case. When u has not rated i (*i.e.*, $r_{ui} = null$), determining p_{ui} becomes non-trivial. Therefore, our main challenge is to accurately infer pre-use preferences p_{ui} when unrated.

To address our challenge, we propose to borrow the framework of the *one-class collaborative filtering* (OCCF) problem [Pan08], [Sin09]. The OCCF problem occurs when a rating score is *unary* such as clicks, bookmarks, and purchases so that either a cell in a matrix has null value or a single value indicating "yes." Meanwhile, ambiguity arises in the interpretation of unrated items. That is, it is difficult to differentiate *negative* and *positive* examples that co-exist among unrated items [Pan08]. Some unrated items could be positive as the user was not aware of the existence of the items but if she knew she would have liked them. On the other hand, some are negative as the user knew about the items but decided not use them as she did not like them.

This problem setting happens when we infer pre-use preferences for unrated items. That is, known pre-use preferences for rated items have a value of 1 (*i.e.*, $p_{ui} = 1$), and missing pre-use preferences for unrated items are ambiguous. In Figure 3.2, we observe that both unlabeled positive examples ($I_u^{in} - I_u^{eval}$) and negative examples (I_u^{un}) co-exist in the set of items whose pre-use preferences are unknown ($I - I_u^{eval}$). We thus employ the OCCF method [Pan08] to infer pre-use preferences.

The basic idea of the OCCF method is to treat all unrated items as negative examples, and assign weights to quantify the relative contribution of these examples. In our situation, it assigns 0 to p_{ui} whose value is null in P and determines weight w_{ui} by three schemes: *uniform*, *user-oriented*, and *item-oriented* schemes. In this chapter, we employ the user-oriented scheme that was the best performer in [Pan08]. The intuition of the user-oriented scheme is essentially *as a user rates more items, she is more likely to dislike unrated items.* That is, it computes the weight w_{ui} in proportion to the number of items rated by u: $w_{ui} = \sum_i p_{ui}$. The OCCF method finally updates the value of p_{ui} whose value is 0 using all entries in P and their corresponding weights. We treat the updated values as the inferred pre-use preference scores.

To update the values, the OCCF method employs the weighted alternating least squares (wALS) method [Sre03] in building an SVD model with a matrix and weights. It infers the preference scores for each user's unrated items via the SVD model. The wALS method decomposes a matrix P into two low-rank matrices X and Y while optimizing an objective function $\mathcal{L}(X,Y)$. The matrix P represents observed pre-use preferences in our case, *i.e.*, $P = (p_{ui})_{m\times n}$. The matrices X and Y represent the features of users and items for latent factors, respectively. The objective function is represented as follows:

$$\mathcal{L}(X,Y) = \sum_u \left[\sum_i w_{ui}\left\{(p_{ui} - X_u Y_i^T)^2 + \lambda\left(\left\|X_{u(\cdot)}\right\|_F^2 + \left\|Y_{i(\cdot)}\right\|_F^2\right)\right\}\right] \tag{3.1}$$

where p_{ui} and w_{ui} are the entries in the observed pre-use preference matrix P and the weight matrix W, respectively. The vector X_u is the u-th row of matrix X, and the vector Y_i is the i-th row of matrix Y. The two vectors represent the features of user u and item i. In addition, $\|\cdot\|_F$ denotes the *Frobenius norm* and λ is a regularization parameter.

In order to factorize the matrix P, the OCCF method first assigns random values to elements in the matrix Y, and updates elements in the matrix X as in Equation (3.2) by optimizing the objective function. $\forall 1 \le u \le m$:

$$X_{u(\cdot)} = p_{u(\cdot)}\widetilde{w}_{u(\cdot)}\mathrm{Y}\left\{Y^T\widetilde{w}_{u(\cdot)}Y + \lambda(\textstyle\sum_i w_{ui})L\right\}^{-1} \tag{3.2}$$

where $\widetilde{w}_{u(\cdot)}$ is a diagonal matrix with elements of $w_{u(\cdot)}$ on the diagonal, and matrix L is an identity matrix. After that, the OCCF method updates elements in the matrix Y while fixing the matrix X as in Equation (3.3). $\forall 1 \le u \le n$:

$$Y_{i(\cdot)} = \mathrm{p}_{(\cdot)i}^T\widetilde{w}_{(\cdot)i}\mathrm{X}\left\{X^T\widetilde{w}_{(\cdot)i}X + \lambda(\textstyle\sum_u w_{ui})L\right\}^{-1} \tag{3.3}$$

We optimize the objective function by repeating Equation (3.2) and Equation (3.3) until matrices X and Y converge to a local optimum. Finally, we approximate matrix $\hat{P}$ by calculating an inner product of X and Y as in Equation (3.4) where an entry $\hat{p}_{ui}$ in the matrix $\hat{P}$ represents a pre-use preference score of user u for item i.

$$\hat{P} \approx \mathrm{P} = \mathrm{X}Y^T \tag{3.4}$$

3.2.2 Identifying Uninteresting Items

Once pre-use preferences of unrated items are inferred, next, we attempt to identify uninteresting

items. Based on the pre-use preference scores inferred by the OCCF method, a user u's uninteresting items are defined as follows:

$$I_u^{un}(\theta) = \{i | \rho(\hat{p}_{ui}) \leq \theta, r_{ui} = null\} \tag{3.5}$$

where $\rho(\hat{p}_{ui})$ indicates the percentile rank of $\hat{p}_{ui}$ among all user-item pairs whose ratings are missing in R. For instance, $I_u^{un}(20)$ indicates that we assign all unrated items whose percentile ranks of pre-use preference scores are at the bottom 20% as uninteresting items.

In Equation (3.5), we do not use an absolute cut-off value for pre-use preference scores because the OCCF method is originally designed for computing users' *relative* preferences. In addition, we adjust the parameter θ to obtain the best accuracy for top-N recommendation. If θ is set high, a large number of zero ratings are injected to Z, leading a less sparse rating matrix at the cost of some zeros being false set. On the hand, if θ is set low, we may not be fully utilizing the benefit of uninteresting items as only a small number of zero ratings are injected. Our simple use of *relative* cut-off based on percentile rank works well, at the end, as we will demonstrate in experiments.

EXAMPLE 2. Figure 3.4 illustrates a pre-use preference matrix $\hat{P}$, where the cells with 1 and with decimal numbers are originally derived from the rated and unrated items in R, respectively. As a larger θ is set, more items are determined as uninteresting items. For example, when θ = 20, only light-colored cells become uninteresting items. If θ = 80, both light and middle-colored cells become uninteresting items. Finally, when θ = 99, all colored cells become uninteresting items. Note that the number of uninteresting items could be different per user. For instance, when θ = 80, the numbers of uninteresting items for u_1 and u_2 are 1 and 4, respectively.

	i_1	i_2	i_3	i_4	i_5	i_6
u_1	1	1	1	1	0.56	0.99
u_2	1	1	0.03	0.52	0.57	0.60
u_3	1	1	0.04	0.53	0.58	0.92
u_4	1	0.01	0.05	0.54	0.59	0.93
u_5	1	0.02	0.51	0.55	0.91	0.94

Uninteresting items: (θ = 20%) + (θ = 80%) + + (θ = 99%)

Figure 3.4. A Pre-use preference matrix $\hat{P}$.

It is worthwhile to emphasize that our approach identifies uninteresting items more broadly than

what a user herself would have recognized. In a real setting, even if asked, users are able to review only a small fraction of (millions of) unrated items to identify truly uninteresting items. In contrast, our approach finds uninteresting items that users have not recognized yet but are likely to consider uninteresting when presented.

3.2.3 Zero-Injection

This chapter proposes a novel approach to fill a part of missing ratings, named as *zero-injection*, which assigns a zero value to r_{ui} in R if an item i is determined as a user u's uninteresting item. We inject zeros for missing ratings because a user would not be satisfied with her uninteresting items even when recommended.

By augmenting a rating matrix R with zeros, the zero-injection method builds a new matrix that contains zero ratings as well as actual user ratings. We call this augmented matrix as *zero-injected matrix* $Z = (z_{ui})_{m \times n}$, where entry z_{ui} can be represented as follows:

$$z_{ui} = \begin{cases} r_{ui} & \text{if } u \text{ has evaluated } i; \\ 0 & \text{if (1) } u \text{ has not evaluated } i, \text{ and} \\ & \text{(2) } i \text{ is an uninteresting item to } u; \\ null & \text{otherwise} \end{cases}$$

Note that z_{ui} is set as r_{ui} if user u has rated item i. On the other hand, if i has not been rated by u ($r_{ui} = null$) and its pre-use preference p_{ui} is not high, z_{ui} is set as 0. Otherwise, z_{ui} is set as *null*, indicating "unknown." Entry z_{ui} indicates a user u's rating for an item i. When the value of z_{ui} is zero, it implies that user u is very unlikely to love item i even though i is recommended to u. When the value of z_{ui} is the rating given by user u, it indicates that user u is satisfied with item i to the extent that its rating score indicates.

Note that our proposed approach works regardless of the choice of underlying CF methods as we can simply replace the original rating matrix R by the zero-injected matrix Z. In other words, our approach is orthogonal to existing CF methods, which is one of strengths in our approach. Businesses and recommender systems can choose any appropriate CF methods suiting their particular settings, yet still use our proposed idea for improving accuracy and running time.

Our proposed approach can improve the CF methods in two aspects. When CF methods are applied, the zero-assigned items for a user are *excluded* from her recommendation list (as ratings are zero). We note that existing CF methods consider *all* items whose ratings are missing as

recommendation candidates. Therefore, the zero-injected matrix prevents a user's uninteresting items[2] from recommendation. In addition, the zero-injected matrix provides a much more number of ratings (including ratings with zero values) than the original rating matrix. Thus, the CF methods equipped with our approach are able to understand relative users' preferences more accurately when they are applied to the zero-injected matrix. Moreover, as the number of recommendation candidates reduces, the computational cost also decreases because the CF methods have only to compute the relative preferences for a small number of items.

Similar to our approach, PureSVD [Cre10] also assigns zero values to missing ratings. Unlike our approach, however, PureSVD has no regard for identifying users' uninteresting items, and fills zero to "all" missing ratings (items $i \in I - I_u^{eval}$ for each user in Figure 2). PureSVD regards user's favorite items as the items with low preferences ($I_u^{in} \in I - I_u^{eval}$ in Figure 2). Unlike this approach, our zero-injection selectively fills only the items in I_u^{un} of low pre-use preference scores with zeros, recognizing a user u's taste more precisely. Moreover, PureSVD cannot explicitly prevent uninteresting items from being recommended. In experiments, we demonstrate the superiority of our approach over PureSVD.

3.2.4. *Why Does Zero-Injected Matrix Help?*

We argue that the zero-injected matrix helps improve the accuracy of any CF methods. To present the ground for our argument, we discuss the effect of our approach when applied to two popular CF methods: the *item-based collaborative filtering (ICF)* [Sar01] and the *SVD-based method (SVD)* [Zha05]. ICF predicts a rating $\hat{z}_{ui}$ for a target item i of a user u by referencing her ratings on those items similar to the item i as follows:

$$\hat{z}_{ui} = \frac{\sum_{j \in S_i} \{z_{uj} * sim(i,j)\}}{\sum_{j \in S_i} sim(i,j)} \tag{3.6}$$

where S_i is a set of (up to) k items that have users' rating patterns most similar to that of i and u's rating is known for. If there are less than k users who have evaluated i, S_i includes this number of items only instead of k. In addition, $sim(i,j)$ denotes the similarity between items i and j in terms of users' rating patterns.

Figure 3.5 illustrates the difference between a rating matrix R and its corresponding zero-injected

[2]We note that the user is highly unlikely to use those items.

matrix Z built by our approach. We observe that, unlike R, Z has extra zero ratings (in boldface) implying users' uninteresting items.

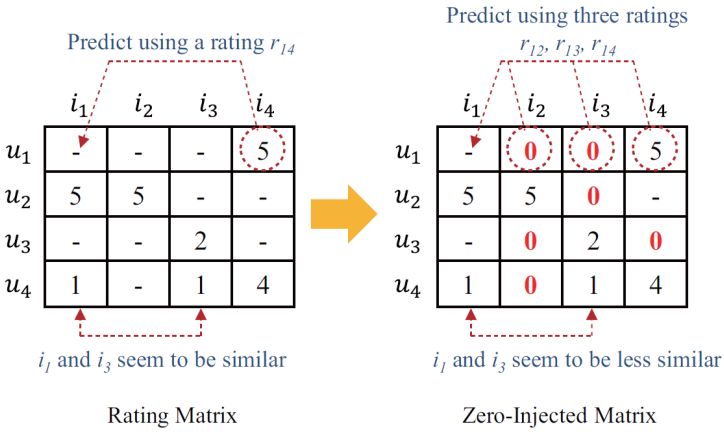


Figure 3.5. Rating and zero-injected matrices in CF.

Suppose that ICF predicts the rating r_{11} of a user u_1 for an item i_1 when its parameter k is set as 3. With the rating matrix, it refers to only r_{14} for prediction[3] because u_1 has not rated i_2 and i_3 (*i.e.*, r_{12} and r_{13} are nulls). Therefore, its predicted rating would be inaccurate due to the sparsity problem. On the other hand, thanks to zero-injection, ICF considers more ratings, z_{12} and z_{13}, in addition to z_{14} with Z. This is because item set S_i now contains a more number of ratings, including two uninteresting items.

In addition, our approach helps find the items that are truly similar. With R, ICF may conclude that i_1 and i_3 are highly similar because u_4 gives 1 to both of them. However, the similarity can be inaccurate because it is based on only a single user's opinion. With Z, the two items are regarded less similar because u_2 rated two items as quite different. As such, the zero-injected matrix is useful to compute a more accurate similarity because it enables CF to reflect more users' opinions. In particular, we note that the zero-injection makes it possible for ICF to successfully find truly similar users who have a set of "uninteresting" items in common, which has been overlooked in existing CF methods.

In Figure 5, we only show the accuracy improvement owing to a more number of items in S_i. Moreover, even when the number of items in S_i is the same, the accuracy could increase because S_i with our zero-injected matrix contains those items more similar to item i than S_i obtained with

[3]Otherwise, the items whose similarity to i_1 is less than i_2 and i_3 need to be included in S_i.

R.

Next, we also explain how the zero-injected matrix makes existing SVD-based methods more accurate [Zha05]. Given *Z*, SVD factorizes it into an inner product of two low-rank matrices *X* and *Y* with a dimension *f*. That is, one low-rank matrix is an *m*-by-*f user-factor* matrix and the other is an *n*-by-*f item-factor* matrix. Each user *u* is thus associated with an *f*-dimensional vector $x_u \in \mathbb{Z}^f$. Each item *i* is involved with an f-dimensional vector $y_i \in \mathbb{Z}^f$. The rating prediction for $\hat{z}_{ui}$ is computed by Equation3.7:

$$\hat{z}_{ui} = x_u y_i^T \tag{3.7}$$

With the original rating matrix *R* in Figure 3.5, SVD cannot recognize that u_2 is related to u_1 or u_3 because they have no common items rated. In contrast, using *Z*, it now successfully observes the relationship between those users, *i.e.*, both u_2 and u_1 are not interested in i_3, and have different opinions on i_2. Similarly, it also misses items' relationships such as i_2 and i_3 with *R* while it is able to find the relationships with *Z*. Therefore, SVD can build a better model representing more latent relationships among users and items with our zero-injected matrix, which helps improve the overall accuracy of top-*N* recommendation.

3.3. Evaluation

In this section, we evaluate the effectiveness and efficiency of our proposed approach with a real-life dataset and an open-source implementation of existing CF methods. Specifically, we first validate the accuracy of the OCCF method [Pan08] to infer users' pre-preferences in comparison with other methods and also verify our assumption that users are unlikely to use uninteresting items with low pre-use preferences. In addition, we perform the sensitivity analysis on parameter θ that is crucial to determine uninteresting items. Finally, we show the accuracy and running time of the modified CF methods equipped with our approach compared with the original ones.

3.3.1. Experimental Set-Up

We use the MovieLens 100K dataset [Res94], widely used for evaluating recommender systems [Pan08, Ste10, Ngu14, Yin15]. This dataset consists of 943 users, 1,682 items, and 100,000 ratings. The ratings are integer values from 1 (*i.e.*, worst) to 5 (*i.e.*, best). The minimum number of ratings

per user is 20.

For evaluating the accuracy of top-*N* recommendation, we vary the value of N from 5 to 20 in an increment of 5 (default value = 5). Only the items with the rating score of 5 are considered as *relevant*, *i.e.*, ground truth, as correctly predicting items with high ratings has more business ramifications (than predicting items with low ratings), as argued in [Ste10].

We adopt four metrics to measure the accuracy such as *precision*, *recall*, *normalized discounted cumulative gain* (nDCG), and *mean reciprocal rank* (MRR). For a user u, precision $P_u@N$ and recall $R_u@N$ can be computed by $\frac{|Rel_u \cap Rec_u|}{|Rec_u|}$ and $\frac{|Rel_u \cap Rec_u|}{|Rel_u|}$, respectively, where Rec_u denotes a set of N items that each method recommends to u, and Rel_u denotes a set of items considered relevant (*i.e.*, ground truth). We also use nDCG to reflect ranked positions of items in Rec_u. Let y_k represent a binary variable for the k-th item i_k in Rec_u. If $i_k \in Rel_u$, y_k is set as 1. Otherwise, y_k is set as 0. Then, $nDCG_u@N$ is computed by $\frac{DCG_u@N}{IDCG_u@N}$, where $DCG_u@N = \sum_{k=1}^{N} \frac{2^{y_k}-1}{log_2(k+1)}$, and $IDCG_u@N$ means an *ideal* $DCG_u@N$ where y_k is set as 1 for every item $i_k \in Rec_u$. MRR shows the average inversed rankings of every item $i_k \in Rec_u$. MRR_u can be computed by $\frac{1}{|Rel_u|}\sum_{i=1}^{|Rel_u|}\frac{1}{rank_i}$. All measurements are averaged using 5-cross validation.

To show the effectiveness of our approach, we first augment a rating matrix R to a zero-injected matrix Z using our proposed method, and feed Z as input to existing CF methods such as an item-based CF (ICF) [Sar01] and SVDbased CF (SVD) [Zha05], well known methods in memory-based and model-based approaches, respectively. Further, we use the ICF and SVD methods implemented in the open-source MyMediaLite [Gan11] using their default parameter settings in our experiments.

Our empirical study is to answer the following questions via comprehensive experiments.

- Q1: Is the OCCF method (most) effective to infer users' pre-use preferences?
- Q2: Are users really not satisfied with uninteresting items that our approach determines?
- Q3: How does the accuracy of our approach vary over different θ? How sensitive is θ?
- Q4: How much does our approach improve existing CF methods with respect to accuracy? Is Hypothesis1 valid (with respect to accuracy)?
- Q5: How much time our approach spends compared to existing CF methods? Is Hypothesis 1 valid (with respect to running time)?

3.3.2. Q1: Inference of Pre-Use Preferences

As the gist of our approach to identify uninteresting items depends on the effectiveness of the OCCF method to infer users' pre-use preferences, first, we test how effective the OCCF method is, against four other candidates (Q1): user-oriented method (UOM), binary item-based method (BIM), item-based method (IM), and random method (RM). UOM borrows the idea from the user-oriented scheme that determines the weights in the OCCF method (mentioned in Section 3.2.1). UOM determines user's pre-use preference scores to be inversely proportional to the number of her rated items. BIM uses the "observed" (binary) pre-use preference matrix P (introduced in Section 3.2.1) and infers the pre-use preference scores exploiting the item-based CF [Sar01]. IM is identical to the original item-based CF, which produces pre-use preference scores based on the original rating matrix R. RM is a baseline that does not infer pre-use preference scores but randomly selects uninteresting items among unrated items. We also employ the same parameter settings as in [Pan08] for wALS in the OCCF method.

As the accuracy measure for Q1, we especially define an *error rate*, which captures how many rated items are selected as uninteresting items (*i.e.*, mis-classified) for each user by an inference method. The idea behind this notion is that as user's pre-use preferences should have been relatively high for her *rated items*, an inference method that does a less number of mis-classification is deemed as a better solution. A user u's error rate is defined as: $err_u^{\theta} = \frac{|I_u^{un}(\theta) \cap I_u^{test}|}{|I_u^{test}|}$, where I_u^{test} a set of items *rated* by u in a test set, and $I_u^{un}(\theta)$ is a set of items determined as uninteresting (*i.e.*, ranked in the bottom θ% according to the inferred pre-use preference scores) by the particular inference method. The lower a user's error rate gets, the better an inference method is.

Figure 3.6 shows the changes of error rates of five inference methods, averaged over all users with varying θ. In general, as θ increases, error rates of five methods increase as well. In particular, the error rates of UOM, IM, and RM increase more rapidly than those of the OCCF method and BIM. The OCCF method and BIM have relatively small error rates until θ reaches to 90%, implying that their accuracy is fairly good when θ is smaller than 90%. Above 90%, their error rates grow rapidly. This is because at this point there are only a relatively small number of unrated items left among which a substantial amount of interesting items exist. Among five methods, the OCCF method shows the best error rates, regardless of θ. Therefore, we conclude that it is the most effective in correctly inferring pre-use preferences.

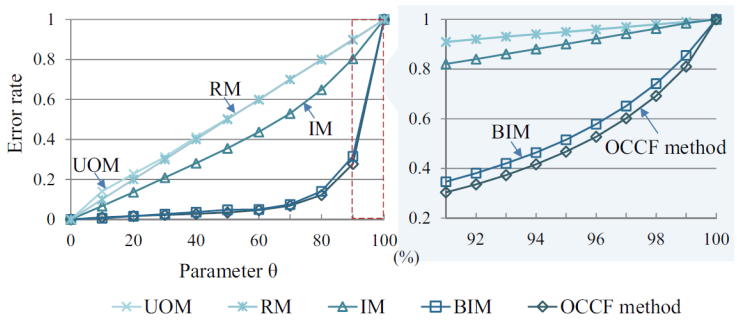


Figure 3.6. Error rate comparison of five inference methods.

So far, we validate that pre-use preferences inferred by the OCCF method are most accurate. Next, we examine if pre-use preferences inferred by the OCCF method indeed yield the best accuracy when used in real CF methods, *i.e.*, the end-to-end performance. We build the zero-injected matrix Z using pre-use preferences inferred by five inference methods, and apply Z to two CF methods–the *item-based CF* (*ICF*) and *SVD-based CF* (*SVD*). In this experiment, we vary θ as 30%, 60%, and 90%. As all accuracy metrics of Section 3.3.1 show similar tendencies, we only report the results for P@5 here (More detailed analysis on the effect of θ will be given in Section 3.3.4).

Table 3.2 shows the precision@5 scores of ICF and SVD using a zero-injected matrix, inferred by five different inference methods. Similar to Figure 3.6, the OCCF method shows the best accuracy in all cases (19%–26% higher than BIM, the second best one) while UOM shows the worst accuracy (even lower than RM). Therefore, as the answer to Q1, we conclude that the OCCF method is the most effective for inferring pre-use preferences, and, in subsequent evaluation, employ only the OCCF method for inferring pre-use preferences.

Table 3.2. Accuracy (i.e., P@5) of CF methods equipped with our approach with five inference methods

CF method	Parameter Θ	Inference Method				
		OCCF	BIM	IM	UOM	RM
ICF	0.3	**0.199**	0.155	0.123	0.062	0.085
	0.6	**0.199**	0.165	0.130	0.021	0.035
	0.9	**0.201**	0.154	0.134	0.004	0.009
Average		**0.200**	0.158	0.129	0.029	0.043
SVD	0.3	**0.177**	0.137	0.110	0.062	0.067
	0.6	**0.189**	0.150	0.144	0.034	0.039
	0.9	**0.207**	0.192	0.144	0.011	0.008
Average		**0.191**	0.160	0.133	0.036	0.038

3.3.3. Q2: User's Satisfaction on Uninteresting Items

One of main assumptions of our proposal is that a user would not be satisfied with her uninteresting items. If our assumption does hold in practice, then a user is unlikely to purchase or use the uninteresting items recommended to her, and even if she does, she is more likely to assign low ratings to these items. To validate this assumption, we examine whether users are going to be satisfied with items in proportion to their pre-use preference scores. Recall that an item is likely to be selected as uninteresting if its pre-use preference score is relatively low.

First, we observe what portion of items are rated according their pre-use preference scores. For this observation, we first hide some ratings (*i.e.*, a test set), and compute the pre-use preference scores for all unrated user-item pairs by using the remaining ratings (*i.e.*, a training set). Next, we divide the unrated user-item pairs into 100 bins according to their percentile rank θ of pre-use preference scores. We calculate the error rates for the *j*-th subset, $I_u^{un}(\beta_j, \beta_{j+1})$, instead of $I_u^{un}(\theta)$ to show the ratio of those user-item pairs that are really rated in the test set. $I_u^{un}(\beta_j, \beta_{j+1})$ includes u's unrated items whose rank ρ is between β_j and β_{j+1} (*i.e.*, $\beta_j \leq \rho(\hat{p}_{ui}) \leq \beta_{j+1}$ for $\forall i \in I_u^{un}(\beta_j, \beta_{j+1})$).

Figure 3.7 depicts the distribution of error rates over ρ. As ρ increases, the error rate increases rapidly. Moreover, the test set verifies that users have evaluated only a few items whose ρ is low. For example, among all the rated items, 90% items (resp. 95% items) have ρ higher than 74% (resp. 79%) (marked in Figure 3.7). This result indicates that users hardly ever use the items whose pre-use preference scores are not high, thereby supporting out assumption.

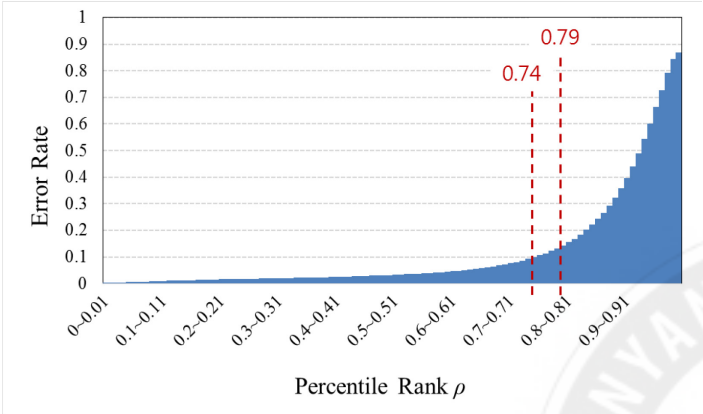


Figure 3.7. Users' error rates according to items' pre-use preference scores.

Next, we investigate users' given rating scores according to their pre-use preference scores. For this analysis, we build training and test sets, and make the OCCF method infer pre-use preferences for unrated user-item pairs by using the training set. After that, we divide those user-item pairs into 10 bins according to their percentile rank ρ of pre-use preference scores. By referring to the test set, we compare the number of items rated as 1 or 2 and that of items rated as 4 or 5 in each bin. For a fair comparison, we need to note two important observations: users leave 4 or 5 ratings (*i.e.*, 55% of all ratings) much more often than 1 or 2 ratings (*i.e.*, 17% of all ratings) in the MovieLens dataset; In addition, the numbers of "rated" items in the test set differ significantly depending on the bins. For example, 0.1% of user-item pairs have ratings in the first bin while 86.8% of pairs do in the 99-th bin. Considering these inequalities, we compute the *relative ratio* of items *rated as 1 or 2* (resp. *4 or 5*) for each bin as follows:

$$ratio_u(j,s) = \frac{|I_u^{test}(s) \cap I_u^{un}(\beta_j,\beta_{j+1})|}{|I_u^{test} \cap I_u^{un}(\beta_j,\beta_{j+1})|} \div \frac{|I_u^{eval}(s)|}{|I_u^{eval}|} \tag{3.8}$$

where I_u^{test} is a set of items evaluated by a user u in the test set, and $I_u^{test}(s)$ indicates a set of items rated as s in I_u^{test}. I_u^{eval} indicates a set of items rated by u among all items (*i.e.*, the test as well as the training set). $I_u^{eval}(s)$ presents a set of items rated as s in I_u^{eval}. The fraction before the division sign in Equation 3.8 means the ratio of items rated as s to all rated items in a bin. In addition, the fraction after the division sign in Equation 3.8 (*i.e.*, the ratio of items rated as s to the whole rated items) is necessary for normalization. A higher relative ratio indicates that more items rated as s exist in a bin.

Figure 3.8 shows the distribution of relative ratios. When ρ is smaller than 0.3, the relative ratio of items rated as 1 or 2 stagnates as ρ gets higher. This is because there are only a few rated items whose ρ is less than 0.3. When ρ is higher than 0.3, the relative ratio of items rated as 1 or 2 decreases as ρ gets higher. When ρ is smaller than 0.9, the relative ratio of items rated as 4 or 5 is smaller than 1. Only when ρ is in the range of 0.9 and 1, the relative ratio is higher than 1. In short, the items whose ρ is smaller than 0.9 are likely to be rated as 1 or 2 rather than 4 or 5. On the other hand, the items whose ρ is higher than 0.9 are more likely to be rated as 4 or 5. Therefore, we know that users are less likely to be satisfied with the items whose ρ is less than 0.9.

Based on the results of two experiments above, as the answer to Q2, we conclude that users are rarely satisfied with the items whose pre-use preferences are not high. In addition, we found that users tend to be unsatisfied with most of items in R (*e.g.*, 90%), suggesting that there are many items uninteresting to a user.

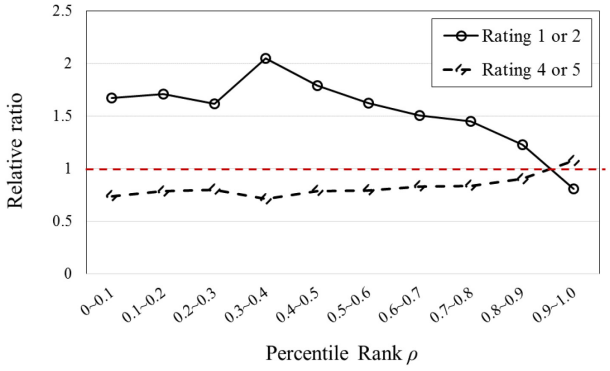


Figure 3.8. Distribution of users' pre-use preference scores for rated items.

3.3.4. Q3: Effect of Parameter θ

As the parameter θ controls the amount of values imputated with zero-injection, it greatly affects the accuracy in recommendation. To verify the effect of θ, therefore, we conduct the sensitivity test. We first build different zero-injected matrices with varying θ and apply them to two CF methods, ICF [Sar01] and SVD [Zha05].

Figure 3.9 shows the accuracy of top-N (N = 5) recommendation with ICF and SVD with varying θ. We increase θ in an increment of 10% for the range of 10−90%, while increase θ in an increment of 1% for two extreme ranges, 0−10% and 90−99.7%. Note that we do not report the result with θ = 100% because CF methods with our approach recommend *nothing* in this case. For this reason, we set θ up to 99.7% in order to leave only 5 items whose pre-use preference scores are the highest for each user. In this case, these 5 remaining items are thus *all* recommended (as top-5) to each user without requiring further CF methods. In summary, the result with θ = 0% indicates the accuracy of original ICF and SVD methods without using our approach, while the result with θ = 99.7% indicates the accuracy of the OCCF method without using ICF or SVD.

In Figure 3.9, we observe that the results of precision, recall, nDCG, and MRR show similar patterns. All these accuracy values of CF methods increase as θ increases up to around 95%. Moreover, accuracies in general grow rapidly until θ reaches 10%. All results clearly show that our idea of using zero-injection *dramatically improves* the accuracy of two original CF methods. ICF using our approach with θ = 96% shows the best precision, *5.2 times higher* than ICF without our approach. When θ = 95%, similarly, our approach improves the precision of SVD by *3.4 times*.

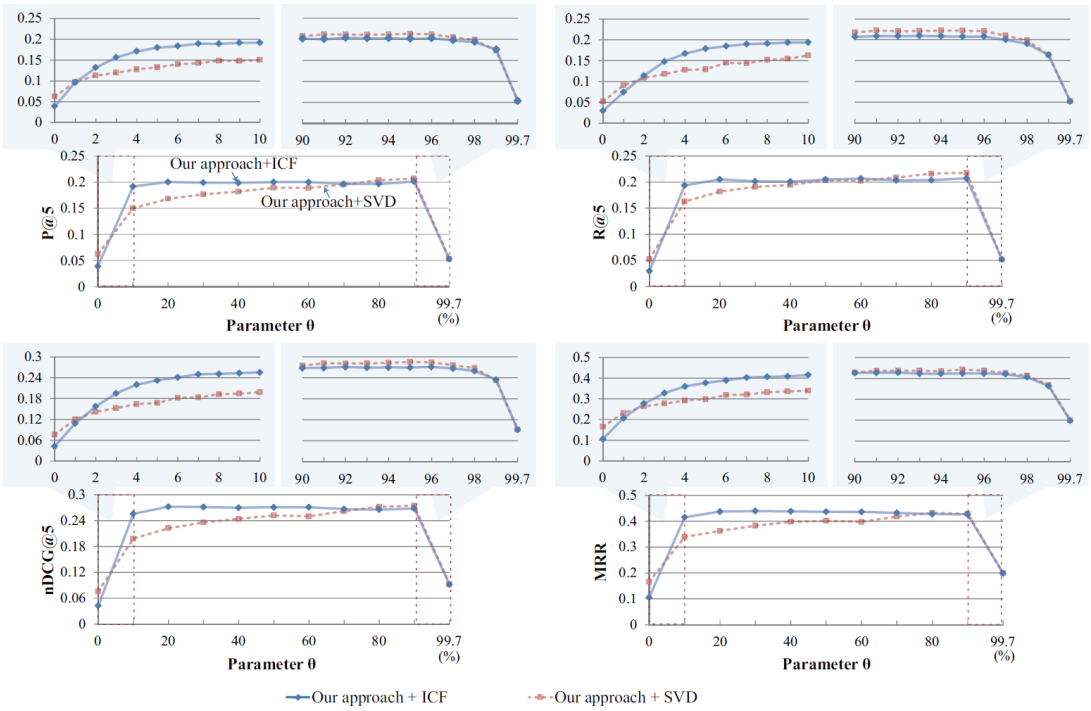


Figure 3.9. Accuracy of ICF and SVD equipped with our proposed approach with varying parameter $\boldsymbol{\theta}$.

As mentioned earlier, the OCCF method can be used to produce top-*N* recommendation without using ICF and SVD (*i.e.*, θ =99.7%). The method, however, shows accuracy much lower than ICF and SVD equipped with our approach. This implies that the OCCF method is quite effective in finding uninteresting items. However, it is not that effective in recommending the final items because it ignores rating scores.

The accuracy changes greatly when θ is less than 10% or more than 90% while it changes little when θ is between 10% and 90%. We can interpret this phenomenon as follows: (1) as θ increases up to 10%, more user-item pairs (*i.e.*, highly likely to be uninteresting) are filled with zeros (giving *accurate* and *useful* information to CF methods) and are also *correctly* excluded from top-*N* recommendation. (2) When θ ranges between 10% and 90%, accuracy changes little because filling unrated user-item pairs in the case of (1) has already alleviated most of the data sparsity problem. Filling more zero ratings no longer gives useful information to CF methods although user-item pairs whose θ is between 10% and 90% are highly likely to be uninteresting (See Section 3.3.3). (3) When θ is larger than 90%, the accuracy decreases significantly. As θ reaches 99.7%, more user-item pairs having *high* pre-use preferences (*i.e.*, could be interesting to users) are

incorrectly filled with zeros (giving inaccurate and less useful information to CF methods) and could be *incorrectly* excluded from top-*N* recommendation.

To understand this phenomenon more clearly, we define five sets of uninteresting items according to the percentile rank ρ of pre-use preference scores. Figure 3.10 shows the accuracy of SVD with the five sets of uninteresting items including those items whose ρ is 0−20, 20−40, 40−60, 60−80, and 80−99.7%, respectively. In the results, all accuracy metrics show similar tendencies. The four results of 0−20, 20−40, 40−60, 60−80% produce good accuracies, while that of 80−99.7% shows significantly worse accuracies. This implies that, among the items whose ρ is 0−80%, there are only a few interesting items (errors) as we explained above. However, among the items whose ρ is 80−99.7%, there are many interesting items. This phenomenon is also observed in Section 3.3.3.

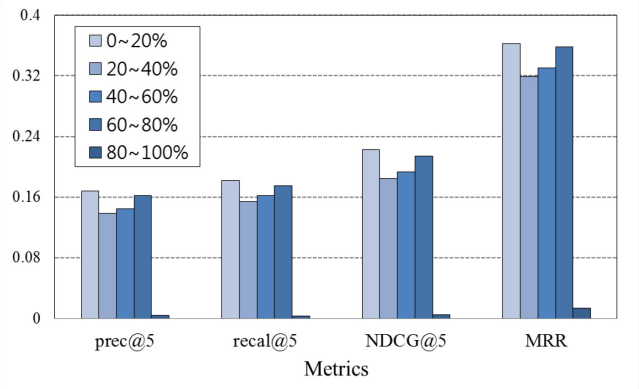


Figure 3.10. Accuracy of SVD with five uninteresting item sets defined by percentile rank $\boldsymbol{\rho}$.

The best value for the parameter θ can be differed according to the dataset, especially on data sparsity, which indicates the proportion of user-item pairs having a rating to the entire number of all pairs. We observe the best parameter θ with several training sets whose sparsity values are different. Figure 3.11 shows the accuracy of SVD and ICF with six sets including 10, 20, 40, 60, 80, and 100% ratings of the training set used in the previous experiments. As more ratings are included in the training sets, the higher precisions SVD and ICF produce. When using 10% and 20% ratings, SVD using our approach with θ = 50% and 70% shows the best precision. Similar to Figure 3-9, the results with using 40, 60, and 80% ratings indicate that SVD produce the best precision when θ is 90%. When using all kinds of training sets, ICF using our approach with θ = 90% shows the best precision. Thus, we claim that our approach improves the accuracy even on sparser dataset, and the best value for parameter θ is not dramatically different according to the sparsity of dataset.

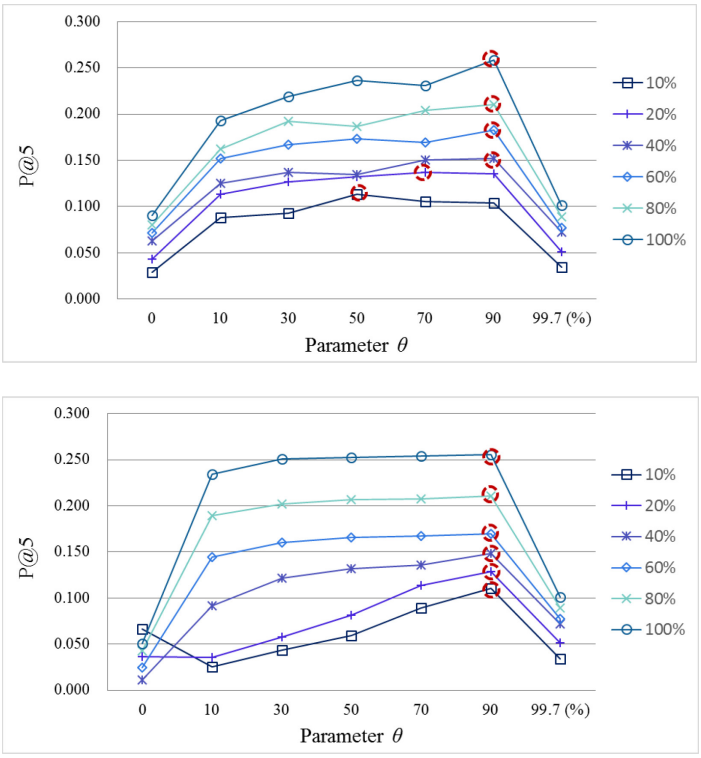


Figure 3.11. Accuracy of ICF and SVD equipped with our proposed approach with varying data sparsity.

As to Q3, finally, we conclude that accuracies increase greatly when θ is less than 10% while it decreases significantly when θ is more than 90%. In addition, accuracies remain quite high and changes little for a large interval of $10\% \leq \theta \leq 95\%$, indicating that we can obtain a high accuracy even if we arbitrarily set θ within that interval, making our approach *parameter-insensitive* (with respect to θ).

3.3.5. Q4: Accuracy of CF Methods with Our Approach

We apply our approach to four existing CF methods (*i.e.*, ICF [Sar01], SVD [Zha05], *SVD++* [Bel07, Kor08], and *pureSVD* [Cre10]), and validate its effectiveness in improving accuracy. SVD++ utilizes both ratings and binary ratings (whether a user evaluates an item or not). PureSVD fills *all* missing ratings with zeros, and then produces recommendation based on the SVD model. Based on the findings in Section 3.3.4, we set the parameter θ as 90%.

Table 3.3 shows the accuracy of all CF methods with and without our approach. We denote a CF

method equipped with our approach as *name_ZI* (*i.e.*, zero-injection) such as ICF_ZI, SVD_ZI, SVD++_ZI, and PureSVD_ZI. The numbers in boldface indicate the highest accuracy among all CF methods with and without our approach.

Table 3.3. Accuracy of Four CF methods equipped with our approach ($\theta = 90\%$)

Metric		ICF			SVD			SVD++			PureSVD		
		Orginal	Ours	Gain	Orginal	Ours	Gain	Orginal	Ours	Gain	Orginal	Ours	Gain
Precision	@5	0.039	**0.201**	**413.8%**	0.063	**0.207**	229.7%	0.076	**0.193**	153.3%	0.100	**0.106**	6.7%
	@10	0.041	**0.161**	**292.6%**	0.056	**0.166**	196.9%	0.069	**0.154**	123.9%	0.082	**0.089**	9.1%
	@15	0.040	**0.137**	**243.7%**	0.053	**0.142**	169.9%	0.063	**0.134**	112.0%	0.071	**0.078**	11.3%
	@20	0.039	**0.121**	**211.7%**	0.048	**0.125**	159.1%	0.058	**0.118**	102.3%	0.063	**0.071**	13.7%
Recall	@5	0.030	**0.207**	**600.3%**	0.052	**0.218**	316.0%	0.063	**0.194**	209.6%	0.112	**0.120**	6.9%
	@10	0.059	**0.305**	**412.7%**	0.089	**0.325**	265.9%	0.109	**0.288**	163.1%	0.175	**0.191**	9.3%
	@15	0.085	**0.375**	**341.4%**	0.121	**0.394**	226.3%	0.150	**0.361**	141.2%	0.220	**0.245**	11.4%
	@20	0.111	**0.428**	**285.4%**	0.144	**0.450**	213.5%	0.184	**0.415**	125.3%	0.254	**0.293**	15.4%
nDCG	@5	0.043	**0.268**	**527.9%**	0.076	**0.274**	260.7%	0.087	**0.256**	196.0%	0.135	**0.143**	6.0%
	@10	0.053	**0.285**	**436.0%**	0.084	**0.297**	252.0%	0.099	**0.272**	175.6%	0.151	**0.162**	7.6%
	@15	0.062	**0.303**	**390.7%**	0.094	**0.315**	234.8%	0.110	**0.291**	163.7%	0.164	**0.178**	8.9%
	@20	0.071	**0.319**	**351.7%**	0.101	**0.332**	227.3%	0.121	**0.306**	153.5%	0.174	**0.193**	10.9%
MRR		0.106	**0.426**	**303.0%**	0.165	**0.428**	159.2%	0.181	**0.416**	129.3%	0.262	**0.274**	4.7%

Among *existing* CF methods, PureSVD has the best accuracy while ICF shows the worst accuracy. In literature, both PureSVD and SVD++ are known to provide a better accuracy than SVD and ICF. We confirmed that our finding is consistent with [Cre10]. We observe that our approach *dramatically improves* the accuracy of all the existing CF methods. For example, our approach improves P@5 of ICF, SVD, and SVD++ by *5, 3.3, and 2.5 times*, respectively. When our approach is applied to PureSVD, its improvement is the smallest. The reason is that PureSVD already assigns zeros to *all* missing ratings, which is an idea similar to the zero-injection in our approach. However, PureSVD fills zeros even for the items that *could be interesting* to users, which could affect the accuracy adversely. Furthermore, our approach employs another strategy of excluding uninteresting items from top-*N* recommendation that contributes to the slight accuracy improvement over PureSVD.

Among the CF methods equipped with our proposed approach, SVD_ZI performs the best, followed by ICF_ZI. Our approach improves both SVD and ICF greatly because they consider all unrated items as unknown ones. For this reason, SVD_ZI and ICF_ZI adopt additional information correctly by regarding zero ratings as uninteresting ones and null values as unknown ones. Meanwhile, our approach cannot improve SVD++ and PureSVD as much as SVD and ICF because SVD++ and PureSVD originally have a positive view on rated items and a negative view on unrated

items. SVD++ builds a SVD model by considering whether a user rates an item, but SVD++_ZI cannot distinguish the uninteresting items and rated items because all of them have rating values. PureSVD_ZI fails to discriminate zero ratings and null values in our zero-injected matrix. Specifically, since it assigns zero ratings again to all unrated items in our zero-injected matrix, in view of rating scores, unrated interesting items (determined by our approach) are also regarded as uninteresting.

We note that SVD_ZI (the most accurate CF method equipped with our approach) has an accuracy about *2 times higher* than PureSVD on average, the best one among existing CF methods, found in our experiments and also reported in [Cre10]. Furthermore, PureSVD_ZI, the least accurate CF method equipped with our approach, still outperform all existing CF methods without using our approach.

In summary, our approach significantly improves the accuracy of all four CF methods used in our experiments via zero-injection while the degrees of improvement vary. As the answer to Q4, therefore, we conclude that our approach improves the accuracy of existing CF methods by *2.5 to 5 times on average, which is significant in comparison with the results obtained by other state-of-the-art methods reported in literature* [Cre10, Ha12].

3.3.6. Q5: Running Time of CF Methods with Our Approach

In this section, we compare the execution times of CF methods with and without our approach. Our approach has both strengths and weaknesses in terms of execution times. We note the weaknesses happen at the pre-computation stage (offline) while the strengths happen at the recommendation stage (online). The CF methods build a model or computes similarities of item pairs during the pre-computation stage; they compute relative preferences and find top-*N* items during the recommendation stage.

Our approach reduces the recommendation time because it significantly reduces the number of candidate items whose relative preferences need to be predicted. Meanwhile, our approach may require more pre-computation time because it has to infer pre-use preference scores for all missing ratings by exploiting the OCCF method. In this section, therefore, we study the trade-off of our approach by examining the running time when our approach is applied to both ICF and SVD.

Figure 3.12 shows both recommendation and precomputation times of SVD_ZI, SVD, SVD++, and pureSVD. Specifically, the recommendation time indicates those for predicting users' ratings

and providing the items to users; the pre-computation time indicates the time for building a SVD model with a rating matrix R (SVD, SVD++, and PureSVD) and a zero-injected matrix Z (SVD_ZI). In Figure 3.12(a), the recommendation time of SVD_ZI decreases rapidly as θ increases because there remain *fewer* candidate items as θ increases. In addition, SVD_ZI takes a shorter time at the recommendation stage. It takes about 1.54 seconds when it has the highest accuracy ($\theta = 90\%$), which is 17% shorter than that of SVD.

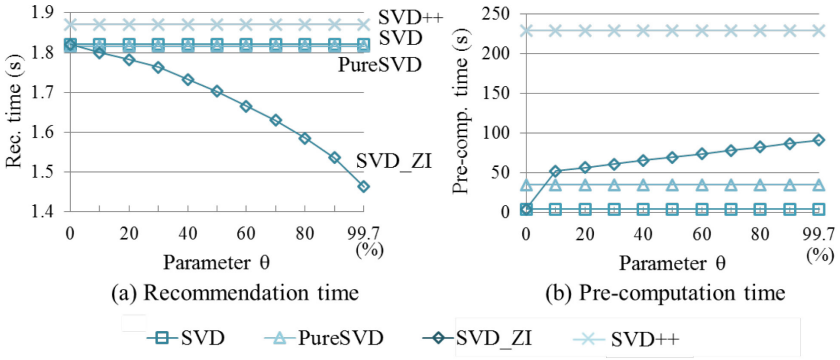


Figure 3.12. Execution time of SVD variants.

In Figure 3.12(b), SVD_ZI takes more time to pre-process items with larger θ, and is slower than SVD and PureSVD. This is because SVD ZI builds two models, one built based on the pre-use preference matrix P and the other based on the zero-injected matrix Z while both SVD and PureSVD build only a single model. SVD_ZI, however, needs less precomputation time than SVD++ that has a more complicated process for building a model.

Figure 3.13 shows both recommendation and pre-computation times of ICF_ZI and ICF. The pre-computation time indicates the time for computing the similarities of all pairs of items. In Figure 3.13(a), when θ is smaller than 20%, the recommendation time of ICF_ZI increases as θ increases. This is because a more number of ratings are used for predicting a rating. It decreases linearly as θ increases when θ is higher than 20%. This is because the number of ratings used for prediction is fixed as k (explained in Section 3.2.4) from this point while there remain a fewer items whose rating is null as θ is set larger. Compared with ICF, ICF_ZI requires less recommendation time when θ is larger than 70%. As we know that the accuracy of ICF_ZI gets higher as θ increases, a user would be satisfied with ICF_ZI when θ is set larger than 70% in terms of both accuracy and recommendation time.

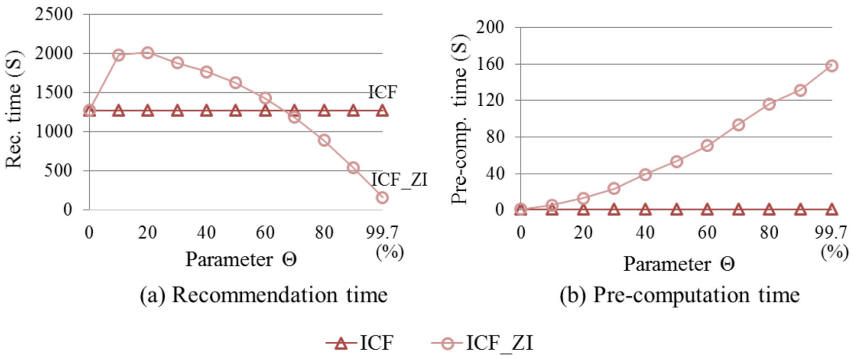


Figure 3.13. Execution time for ICF variants.

Figure 3.13(b) shows the pre-computation time for computing similarities between items [Sar01]. The pre-computation time of ICF_ZI increases as θ increases because a more number of ratings need to be compared to compute similarities of a pair of items. Therefore, ICF_ZI requires more pre-computation time than ICF does.

In summary, our approach reduces the recommendation times of SVD and ICF with $\theta > 70\%$ while it needs more precomputation time. Considering that online recommendation time is more crucial than offline pre-computation time, we strongly believe that our approach improves those CF methods in terms of running times. As the answer to Q5, therefore, according to Figure 3.9, 3.11, and 3.12, the CF methods equipped with our approach produce *more accurate* results in a *shorter time* when θ is set higher than 70%.

4. Exploiting Trustors in the Trust Network

4.1 Existing Approaches

There have been approaches that predict the ratings of an active user *u* on unseen items by utilizing the ratings information of her trustable users on that item. TidalTrust [Gol05], MoleTrust [Mas07], and TrustWalker [Jam09] are typical examples of trust-based recommendation methods. In these methods, the set of trustable users for an active user *u* corresponds to the set $FWD(u)$ in our definition above.

TidalTrust predicts the rating for an active user *u* on an item *i*, $\hat{r}_{u,i}$, by Equation (4.1) below.

$$\hat{r}_{u,i} = \frac{\sum_{v \in FWD(u)} t_{u,v} r_{v,i}}{\sum_{v \in FWD(u)} t_{u,v}} \tag{4.1}$$

Here, $r_{v,i}$ is the rating that a user *v* gave on the item *i*, and the weight $t_{u,v}$ is the trust value by which user *u* assessed user *v* in the trust network. If *u* did not assess *v*, TidalTrust infers $t_{u,v}$ by aggregating the trust values between *u*'s direct trustees and *v*, weighted by the trust values of *u* and her direct trustees.

Similarly, MoleTrust predicts $\hat{r}_{u,i}$ by Equation (4.2).

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{v \in FWD(u)} t'_{u,v} (r_{v,i} - \bar{r}_u)}{\sum_{v \in FWD(u)} t'_{u,v}} \tag{4.2}$$

Here, $\bar{r}_u$ is the average value of all the ratings that user *u* gave on items, and $t'_{u,v}$ is the trust value by which user *u* assessed user *v* in the trust network. Note, however, that MoleTrust computes $t'_{u,v}$ differently from TidalTrust for the case *u* did not assess *v*; it aggregates the trust values between *u* and those users who directly trust *v*, weighted by the trust values. Other symbols are defined as the same as in Equation (4.1).

Unlike the two methods above, TrustWalker does not use trust values but employs a *random walk model* [Bri98], where a walker moves around edges randomly on a network. In this method, the rating on item *i* for user *u* is predicted using two kinds of probability. The first one is the probability that a random walker starting from node *u* stops at node *v* on trust network; the probability is proportional to the similarity between *u* and *v* in terms of their ratings. The second one is the probability that the walker selects item *j* among the items that have been rated by user *v*; the probability is proportional to the similarity of *i* and *j*. As a result, TrustWalker predicts $\hat{r}_{u,i}$ by

Equation (4.3).

$$\hat{r}_{u,i} = \sum_{\{(v,j)|R_{v,j}\}} P\left(XY_{u,i} = (v,j)\right) r_{v,j} \tag{4.3}$$

where $R_{v,j}$ is a boolean variable indicating whether user v gave a rating on item j, and $XY_{u,i} = (v,j)$ represents the combination of the two kinds of probability explained above. Note that these methods only take $FWD(u)$ into account, but not $BWD(u)$ and $UND(u)$. In this chapter, we will investigate the effect of applying the latter two sets to the trust-based recommendation.

4.2 Comparisons of Trustable User Sets

As introduced already in Section 1.3, given an active user u, we define three sets of trustable users in a trust network: $FWD(u)$, $BWD(u)$, and $UND(u)$. $FWD(u)$ is the set of users reachable from u by following the links of trust relationship in the forward direction, $BWD(u)$ the set of users reachable from u by following the links in the backward direction, and $UND(u)$ the set of users reachable from u in the corresponding undirected graph of the trust network.

To understand the similarity and the difference among these three sets rigorously, we performed an empirical study using real-world data sets obtained from Epnions.com [Mas04], a popular online product review site. The Epinions data sets contain users and items such as movies and books. Users are invited to give ratings on items to indicate how much they liked an item by choosing an integer in the range from 1 to 5. In addition, users are also invited to build a trust relationship with other users in the system by asserting binary trust statements (e.g., 1 means 'trust' and 0 'do not trust').

In this study, we used two Epinions data sets which are used in [Mas07, Ma08,Yu11, Ma09a, Mog11], and Table 4.1 shows their statistics. The first data set (Epinions1) contains 167K users, 776K reviews, 13,668K ratings about the reviews, and 718K trust statements among the users; the second (Epinions2) contains 49K users, 140K items, 665K ratings about the items, and 487K trust statements. Note that the ratings in Epinions1 are about product reviews, whereas the ratings in Epinions2 are about products. For both data sets, we regard those users who gave ratings on less than 6 items as *cold-start users*. Both data sets are quite sparse in that cold-start users occupy about 43% and 37% of the total, with the sparsity ratio 0.00011 and 0.000097 for Epinions1 and Epinions2, respectively. The sparsity ratio indicates the proportion of user-item pairs having a rating to the entire number of all possible pairs.

Table 4.1. Statistics of the Epinions data sets used

Content	Data sets	
	Epinions1	**Epinions2**
Users	167,406	49,289
Reviews (or Items)	775,760	139,738
Ratings	13,668,320	664,824
Trust statements	717,667	487,002
Cold-start users	71,078	16,910

When forming the three sets, *FWD*(*u*), *BWD*(*u*), and *UND*(*u*), for an active user *u*, most existing approaches used distance range 4 to 6 from the active user [Gol05, Mas07, Jam09]. In our study, various distance ranges from 1 to 6 were used to analyze and compare the resulting sets.

Table 4.2 shows the average number of members in the three sets of trustable users for all users in the network with different distance ranges. Obviously, the more members will be included in each set as we increase the distance range. We observe a dramatic increase in numbers when the distance range changes from 4 to 5. Note that more than 99% of the entire users are included in *UND*(*u*) for the case of Epinions2 when the range becomes 6, whereas only 36% are included for the case of Epinions1. This is because the trust network used in Epinions2 is much denser than that of Epinions1.

Note also that the average numbers of users in *FWD*(*u*) and *BWD*(*u*) are equal for all distance ranges. This is true because even if each individual set of *FWD*(*u*) may be different from *BWD*(*u*) for a particular user *u*, the average size of all such sets for all users should be the same.

Table 4.2. The average number of users in the three sets of trustable users

Distance range	Epinions1			Epinions2		
	FWD(u)	**BWD(u)**	**UND(u)**	**FWD(u)**	**BWD(u)**	**UND(u)**
1	4.3	4.3	7.1	9.9	9.9	15.5
2	166.4	166.4	680.4	399.9	399.9	1277.8
3	2059.0	2059.0	10617.0	4386.4	4386.4	13625.0
4	9492.9	9492.9	37482.0	16334.0	16334.0	36556.0
5	19067.0	19067.0	54816.0	27150.0	27150.0	47180.0
6	23553.0	23553.0	59509.0	31208.0	31208.0	49065.0

Table 4.3 compares the average similarity between an active user and the users in the three trustable-user sets formed as above. When computing the average similarity, we used the *Pearson*

correlation coefficient between the active user and each user in a trustable-user set, over given ratings on the same items, by Equation (4.4).

$$\text{corr}(u,v) = \frac{\sum_{i=1}^{k}(r_{u,i}-\bar{r}_u)(r_{v,i}-\bar{r}_v)}{\sqrt{\sum_{i=1}^{k}(r_{u,i}-\bar{r}_u)^2 \sum_{i=1}^{k}(r_{v,i}-\bar{r}_v)^2}} \quad (4.4)$$

where k is the number of items rated by both users u and v, $r_{u,i}$ the rating given by user u on item i, and $\bar{r}_u$ the average of ratings given by user u. The resulting values of the Pearson correlation coefficient are in the range of [-1, 1], where -1 indicates that the ratings of two users are completely different, and 1 that the ratings of two users are absolutely the same. Note that the similarity score can be computed by the Pearson correlation coefficient only when there exist at least two common items rated by both users. We ignored the cases where similarities are 0 or less than 0.

The results in Table 4.3 suggest that the two trustable-user sets, *BWD*(u) and *UND*(u), would induce almost the same results as *FWD*(u), i.e., the existing approaches. This results support our hypothesis that a trustor and a trustee have similar interests, and hence using *BWD*(u) or *UND*(u) in the prediction would become as effective as using *FWD*(u). The results also suggest that the accuracy of prediction utilizing *UND*(u) would not be decreased even though the number of members in *UND*(u) is much larger than that of *FWD*(u).

Table 4.3. Similarity of interests between an active user and her trustable users

Distance range	Epinions1			Epinions2		
	FWD(u)	BWD(u)	UND(u)	FWD(u)	BWD(u)	UND(u)
1	0.682	0.677	0.685	0.667	0.679	0.673
2	0.717	0.718	0.727	0.671	0.678	0.678
3	0.741	0.741	0.749	0.679	0.684	0.686
4	0.758	0.751	0.754	0.685	0.688	0.691
5	0.763	0.753	0.752	0.689	0.691	0.694
6	0.764	0.751	0.752	0.691	0.693	0.694

Table 4.3 shows the results different from our assumption that the greater the distance between two users in a trust network, the more different their preferences are. This is because more users who frequently evaluated popular items (*i.e.*, also evaluated by many users) are included in the trustable-user set with larger distance range, and these users are likely to be similar to other users. Specifically, those users are likely to be evaluated items with other users, and the given ratings are likely to be also similar. For example, in Epinions2, the values of 74.5% ratings are 4 or 5. To clarify

this, Figure 4.4 shows the number of users whose evaluated items has 100 ratings on average in each trustable-user set on Epinions2. Figure 4.4 shows that there are more those users in the trustable-user set with the greater distance range. It means the more users who are likely to be similar to *u* are included; therefore, the similarity between *u* and the trustable-user set would be high.

Table 4.4. The number of users whose evaluated items has 100 ratings on average in trustable users

Distance range	Epinions2		
	FWD(u)	BWD(u)	UND(u)
1	4.3	3.6	5.1
2	167.0	143.3	446.0
3	2068.0	1556.3	4578.1
4	7485.5	5453.5	11146.7
5	11767.6	8589.0	13753.5
6	13191.7	9685.5	14184.3

Table 4.4 shows the number of users whose trustable-user set is always empty, regardless of distance range. Since any method cannot predict the rating for a user whose trustable-user set is empty, the number of such users will result in low coverage. Especially, we pay special attention to the cold-start users.

According to the table, 79,227 users have empty $FWD(u)$ in Epinions1 while 15,337 users do in Epinions2. No method can predict ratings for these users, but we can reduce the numbers of unpredictable users by utilizing $UND(u)$. In Epinions2, for example, all users except one have at least one turstor. Unlike Epinions2, a large number of users in Epinions1 have empty $UND(u)$ because the users with empty $BWD(u)$ outnumber the users with empty $FWD(u)$. Nevertheless, it became possible for additional 26,287 users to get ratings prediction when $UND(u)$ is used instead of $FWD(u)$. Overall, the table indicates that in Epinions1, about 33% of users with empty $FWD(u)$ can now get ratings prediction by utilizing $UND(u)$, and in Epinions2, about 100% of users with empty $FWD(u)$ can. Thus, we conclude that the coverage can be increased by using $UND(u)$.

The table also reveals that more than half of the users with empty $FWD(u)$ are cold-start users. According to Table 4.1, the number of cold-start users is less than half of the entire users in both data sets Epinions1 and Epinions2. Thus, we believe that cold-start users cause a bigger problem than other users in terms of the coverage because they are more likely to have no trustable users.

In addition, we also checked how similar the three sets are by themselves, especially between the

members in $FWD(u)$ and the members in $BWD(u)$ or $UND(u)$. Although $BWD(u)$ and $UND(u)$ are formed differently from $FWD(u)$ as they use the backward links, in fact it is possible that these three sets consist of almost the same set of users. Figure 4.1 shows some examples that $FWD(U_1)$, $BWD(U_1)$, and $UND(U_1)$ consist of exactly the same set of users given an active user U_1. In Figure 4.1 (a), U_1 trusts the other users, and each of them also trusts U_1 at the same time. Thus, $FWD(U_1)$, $BWD(U_1)$, and $UND(U_1)$ consist of exactly the same set of users, $\{U_2, U_3, U_4, U_5\}$. Figure 4.1 (b) shows a case where the trust links are circular among the three users, and the three sets consist of exactly the same set of users, $\{U_2, U_3\}$, for the active user U_1. When a data set contains many examples of such cases where $FWD(u)$, $BWD(u)$, and $UND(u)$ are almost identical, the coverage will not be increased much by using $UND(u)$. For this reason, we conduct an experiment to examine and compare the constituent members of the three trustable-user sets.

Table 4.5. The number of users whom the trust based recommendation methods recommend using the three sets of trustable users

Who be recommended	Epinions1			Epinions2		
	FWD(u)	BWD(u)	UND(u)	FWD(u)	BWD(u)	UND(u)
All users	79,227	97,507	52,940	15,337	1	1
Cold-start users	48,435	56,211	43,252	10,852	0	0

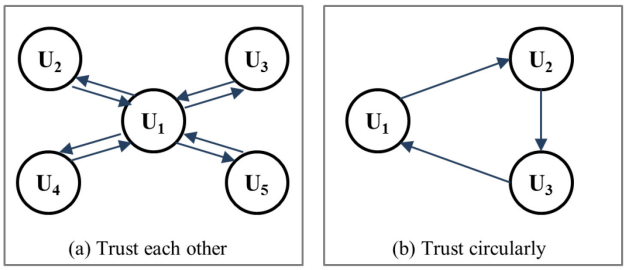


Figure 4.1. Examples of three trustable-user sets containing the same set of users.

For comparison of the three sets, we use the Jaccard coefficient which is a measure to compare the entities of two given sets.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{4.5}$$

Table 4.6 shows the average values of the Jaccard coefficients between pairs of the trustable-user sets. To state the conclusion first, the figures in the table indicate that for an active user u, $BWD(u)$

contains members not appearing in $FWD(u)$, and so does $UND(u)$. This is simply true because the trustor-trustee relationship is not symmetric.

The Jaccard coefficients in the table may suggest a tendency that for the distance ranges over 3, more members will commonly appear in the compared pairs ($FWD(u)$ and $BWD(u)$, or $FWD(u)$ and $UND(u)$) as we increase the distance range. Note also that, for the distance ranges over 3, the Jaccard coefficients for Epinions2 are larger than those for Epinions1. This is due to the fact that the trust network of Epinions2 is smaller and denser than that of Epinions1. Since $BWD(u)$ and $UND(u)$ identify new trustable users different from those in $FWD(u)$, we envisage that utilization of $BWD(u)$ or $UND(u)$ can induce results of ratings prediction differently from existing approaches. First, $BWD(u)$ can contain a large number of members different from $FWD(u)$, thus utilizing $BWD(u)$ in the ratings prediction may well produce different results. Second, $UND(u)$ is a superset of $FWD(u)$ and $BWD(u)$, thus utilizing $UND(u)$ will definitely increase the coverage of prediction.

Table 4.6. Jaccard coefficients presenting how many users are included between pairs of trustable-user sets

Distance range	Epinions1		Epinions2	
	FWD(u) vs. BWD(u)	FWD(u) vs. UND(u)	FWD(u) vs. BWD(u)	FWD(u) vs. UND(u)
1	0.132	0.655	0.291	0.531
2	0.064	0.350	0.166	0.311
3	0.071	0.214	0.179	0.238
4	0.106	0.248	0.297	0.387
5	0.124	0.350	0.385	0.561
6	0.131	0.404	0.421	0.634

On the other hand, we compared the results of ratings prediction using the three trustable-user sets. The comparison was done with respect to the items rated so that we can check whether the coverage of prediction has actually been increased or not, as it may not be affected much if many users gave ratings only on the similar set of items, even though the number of members in the trustable-user sets, say $UND(u)$, became larger.

Figure 4.2 shows an example case where the members of all the three trustable -user sets give ratings on the same set of items. In the figure, the solid and the dotted line represent trust relationships and ratings, respectively. Given an active user U_1, in this example, $FWD(U_1)=\{U_2, U_3\}$, $BWD(U_1)=\{U_4, U_5\}$, and $UND(U_1) = \{U_2, U_3, U_4, U_5\}$, and only the three items {2, 3, 4} constitute the predictable sets for each of $FWD(U_1)$, $BWD(U_1)$ and $UND(U_1)$. That is, the same set of items

can only be predicted even though $UND(U_1)$ contains more members than $FWD(U_1)$ and $BWD(U_1)$, because a method can at best predict those items for which the trustable users have given ratings already. In the experiment, therefore, we need to check with our data sets, Epinions1 and Epinions2, whether the users in $UND(u)$ have actually rated more items, and the coverage has been increased as the result.

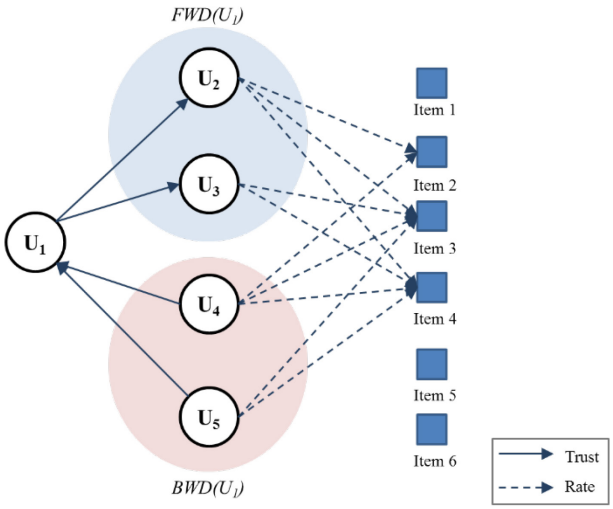


Figure 4.2. Example case where all trustable-user sets rate the same set of items.

In this experiment, the comparison was done again using the Jaccard coefficients, as summarized in Table 4.7. This time, the table suggests a tendency that for the distance ranges over 2, more members will commonly appear in the compared pairs ($FWD(u)$ and $BWD(u)$, or $FWD(u)$ and $UND(u)$) as we increase the distance range. Note also that the values of the Jaccard coefficient in Table 4.7 are generally larger than those in Table 4.6, indicating that many users are interested in common items. In any cases, difference exists with respect to rated items among different trustable-user sets, hence different prediction results will be derived from $BWD(u)$ or $UND(u)$. Consequently, the result of using $UND(u)$ will definitely have better coverage than that of using $FWD(u)$ or $BWD(u)$.

Table 4.8 shows the average number of ratings in the three trustable-user sets. As a matter of fact, the number of ratings grows as we increase the distance range – this is true for all three sets in both Epinions1 and Epinions2, simply because more users are included within larger distance range. For the same reason, the numbers of ratings in the $UND(u)$ sets are generally much larger than those in their corresponding $FWD(u)$ or $BWD(u)$ sets. In Epinions1, the numbers of ratings in $FWD(u)$ are larger than those in $BWD(u)$ for all ranges, whereas in Epinions2 the situation is reversed for the

distance ranges over 3. From this table, we would argue that adding more users to a trustable-user set can increase the coverage because different users are likely to give ratings to different sets of items.

Table 4.7. Jaccard coefficients presenting how many identical items are evaluated by users in pairs of trustable-user sets

Distance range	Epinions1		Epinions2	
	FWD(u) vs. BWD(u)	FWD(u) vs. UND(u)	FWD(u) vs. BWD(u)	FWD(u) vs. UND(u)
1	0.134	0.716	0.271	0.568
2	0.113	0.579	0.186	0.432
3	0.257	0.643	0.343	0.488
4	0.339	0.706	0.537	0.597
5	0.356	0.721	0.605	0.644
6	0.358	0.723	0.620	0.655

Table 4.8. Average number of ratings given by users in the trustable-user sets

Distance range	Epinions1			Epinions2		
	FWD(u)	BWD(u)	UND(u)	FWD(u)	BWD(u)	UND(u)
1	21,763	8,416	25,258	1,016	655	1,300
2	615,653	226,677	1,084,055	32,009	20,374	65,645
3	3,047,153	1,536,985	5,300,759	169,608	157,184	356,604
4	5,026,431	3,539,766	7,551,348	333,518	405,632	594,880
5	5,652,581	4,268,879	7,969,736	411,827	536,527	655,891
6	5,776,576	4,404,631	8,032,498	432,349	570,142	663,961

In this section, we have performed several experimental studies to address the first question mentioned in Section 1.3. That is,

- Do the trustable users in the three sets, $FWD(u)$, $BWD(u)$, and $UND(u)$, have actually similar interests to user u? Will these three sets lead to similar or different results of recommendation?

Through various results discussed in this section, we conclude that users in $BWD(u)$ or $UND(u)$ have similar interests to the active user u, that using $BWD(u)$ can induce prediction results different

from those obtainable using *FWD*(*u*), and that using *UND*(*u*) can improve the coverage of prediction.

4.3 Recommendation Results Using Trustable-User Sets

In order to assess how much improvement can be made by utilizing the two trustable-user sets, *BWD*(*u*) and *UND*(*u*), we performed an experiment of applying these sets to existing methods and measured the coverage and the accuracy of prediction. The chosen methods are TidalTrust [Gol05], MoleTrust [Mas07], and TrustWalker [Jam09]. For each method, the ratings prediction was performed for active user *u* using the three sets, *FWD*(*u*), *BWD*(*u*) and *UND*(*u*). In this experiment, the distance range was set to 6 for all cases, following the idea of "six-degrees of separation" [Mil67].

For measuring the accuracy, we use mean average errors (MAE) and root mean squared errors (RMSE), typically used by existing approaches [Ado05, Jam09, Bre98, Her04]. MAE calculates the mean of the errors (the differences) between predicted ratings and true ratings, whereas RMSE counts larger errors more strongly by magnifying them. MAE and RMSE are computed by Equations (4.6) and (4.7), respectively.

$$MAE = \frac{\sum_{(u,i)|R_{u,i}} (r_{u,i} - \hat{r}_{u,i})}{|\{(u,i)|R_{u,i}\}|} \tag{4.6}$$

$$RMSE = \sqrt{\frac{\sum_{(u,i)|R_{u,i}} (r_{u,i} - \hat{r}_{u,i})^2}{|\{(u,i)|R_{u,i}\}|}} \tag{4.7}$$

In Equations (4.6) and (4.7), $R_{u,i}$ is a boolean variable indicating whether user *u* has a rating on item *i* in the data set, and $r_{u,i}$ and $\hat{r}_{u,i}$ denote the true and predicted ratings, respectively. For the purpose of training and testing, we selected one user-item pair into the test set, following the leave-one-out cross validation [Gol05, Mas07, Sar01, Her04].

The coverage can now be defined as the ratio of the number of recommendable pairs to the total number of user-item pairs used in testing. Additionally, we will also show the results using F-measure that combines RMSE and coverage as defined by Equation (4.8) [Jam09].

$$F - Measure = \frac{2 \times \left(1 - \frac{RMSE}{4}\right) \times Coverage}{\left(1 - \frac{RMSE}{4}\right) + Coverage} \tag{4.8}$$

Table 4.9 shows the resulting values of coverage, accuracy (in MAE and RMSE), and F-measure, computed for the Epinions1 data by applying the three trustable-user sets to the three existing methods. Note that the cases using *UND*(*u*) showed better coverage than those using *FWD*(*u*) for all

the three methods. This improvement in coverage is quite obvious because, for every user u, the set $UND(u)$ is in fact a superset of $FWD(u)$. On the other hand, the coverage became worse in case of $BWD(u)$ compared to $FWD(u)$. This is due to the fact that the number of users whose $BWD(u)$ sets are too small to predict ratings are larger than the number of users whose $FWD(u)$ sets are too small. In Epinions1, for example, 97,507 users have empty $BWD(u)$ sets and 79,227 users have empty $FWD(u)$ sets (see Table 4.5). Trust-based ratings prediction is infeasible in both cases. Note also that the coverage has not actually been increased by $UND(u)$ in case of TrustWalker. We believe that this is because TrustWalker does the prediction using not only ratings (of trustable users) on the target item but also ratings on other items.

Unlike the coverage, the accuracy measures do not change much over different sets of trustable users. In the cases for TidalTrust and MoleTrust, the prediction accuracy using $BWD(u)$ or $UND(u)$ shows results a little better than using $FWD(u)$. In the case for TrustWalker, however, the prediction accuracy using $BWD(u)$ shows results better than using $FWD(u)$ and $UND(u)$. As for F-measure, the cases using $UND(u)$ show the best results for all the three methods.

Table 4.9. Coverage and accuracy with Epinions1 (Case: all users)

Method	User sets	Coverage	MAE	RMSE	F-measure
TidalTrust	FWD(u)	77.6%	0.388	0.762	0.792
	BWD(u)	63.2%	0.378	0.758	0.710
	UND(u)	**82.6%**	**0.367**	**0.733**	**0.822**
MoleTrust	FWD(u)	77.1%	0.472	0.709	0.796
	BWD(u)	62.8%	0.461	0.699	0.714
	UND(u)	**81.9%**	**0.457**	**0.696**	**0.823**
TrustWalker	FWD(u)	85.0%	0.318	0.715	0.836
	BWD(u)	75.7%	**0.261**	**0.675**	0.792
	UND(u)	**85.1%**	0.292	0.694	**0.838**

Table 4.10 shows the results for the Epinions2 data. Again, the cases using $UND(u)$ showed better coverage than those using $FWD(u)$ for all the three methods. Unlike Epinions1, however, this time $BWD(u)$ also induced better coverage than $FWD(u)$. This is due to the fact that in Epinions2 only one user has empty $BWD(u)$, whereas 15,337 users have empty $FWD(u)$ (see Table 4.5).In the case for TidalTrust, the prediction accuracy using $BWD(u)$ and $UND(u)$ shows results slightly better than using $FWD(u)$. In the cases for MoleTrust and TrustWalker, however, the prediction accuracy using

FWD(*u*) shows results better than using *BWD*(*u*) and *UND*(*u*). As for F-measure, the cases using *UND*(*u*) show the best results for all the three methods.

To summarize the results of the two tables above, we conclude that using *UND*(*u*) induced the best coverage for all the existing methods, while maintaining similar levels of accuracy to the cases using *FWD*(*u*) or *BWD*(*u*). In short, utilizing the users linked in a trust network to the maximum (i.e., *UND*(*u*)) improves the coverage of prediction.

Table 4.10. Coverage and accuracy with Epinions2 (Case: all users)

Algorithms	User sets	Coverage	MAE	RMSE	F-measure
TidalTrust	FWD(u)	76.3%	0.873	1.221	0.727
	BWD(u)	87.3%	0.867	1.221	0.774
	UND(u)	**88.2%**	**0.860**	**1.198**	**0.781**
MoleTrust	FWD(u)	75.8%	**0.825**	**1.085**	0.743
	BWD(u)	86.2%	0.836	1.102	0.788
	UND(u)	**87.1%**	0.833	1.097	**0.792**
TrustWalker	FWD(u)	62.8%	**0.897**	**1.270**	0.654
	BWD(u)	63.4%	0.905	1.304	0.653
	UND(u)	**70.3%**	0.907	1.285	**0.691**

Finally, let us look at the cases with cold-start users. Table 4.11 shows the results of coverage, accuracy, and F-measure for the cold-start users with Epinions1, obtained by applying the three trustable-user sets to the three existing methods. Again, the cases using *UND*(*u*) induced better coverage than those using *FWD*(*u*) while maintaining similar accuracy. The cases using *UND*(*u*) also induced higher F-measure values than those using *FWD*(*u*), due to the increased coverage and comparable accuracy. Note that as a matter of fact, the coverage and accuracy values for cold-start users are generally much lower than the corresponding values for the entire users (see Table 4.9). This is because cold-start users not only have little ratings but also have little connections with other users in the trust network.

Table 4.12 shows the results for the Epinions2 data for cold-start users. As for the accuracy, the results show similar tendency to the case for all users (see Table 4.10). As for the coverage, *BWD*(*u*) and *UND*(*u*) induced much better coverage than *FWD*(*u*). Again, this is because, in Epinions2, few users have empty *BWD*(*u*) sets while a large number of users have empty *FWD*(*u*). Moreover, the

trust network of Epinions2 is smaller and denser, hence it is highly probable for a cold-start user to get connected to almost all users within the distance range of 6, even if she had only one trustor. For this reason, the coverage for Epinion2 gets generally higher than Epinion1 (see Table 4.11). As for the F-measure, the results using $BWD(u)$ and $UND(u)$ are higher than those using $FWD(u)$ due to the higher coverage with comparable accuracy.

Table 4.11. Coverage and accuracy with Epinions1 (Case: cold-start users)

Algorithms	User sets	Coverage	MAE	RMSE	F-measure
TidalTrust	FWD(u)	32.8%	0.552	0.976	0.457
	BWD(u)	20.8%	0.588	1.035	0.325
	UND(u)	**43.0%**	**0.542**	**0.971**	**0.549**
MoleTrust	FWD(u)	26.7%	0.749	1.086	0.391
	BWD(u)	16.6%	0.777	1.124	0.270
	UND(u)	**34.1%**	**0.740**	**1.082**	**0.465**
TrustWalker	FWD(u)	33.1%	**0.438**	**0.930**	0.462
	BWD(u)	24.6%	0.454	0.972	0.371
	UND(u)	**41.8%**	0.482	0.981	**0.538**

Table 4.12. Coverage and accuracy with Epinions2 (Case: cold-start users)

Algorithms	User sets	Coverage	MAE	RMSE	F-measure
TidalTrust	FWD(u)	51.8%	0.900	1.276	0.589
	BWD(u)	91.6%	0.896	1.279	0.781
	UND(u)	**92.9%**	**0.886**	**1.250**	**0.790**
MoleTrust	FWD(u)	41.5%	**1.056**	**1.409**	0.506
	BWD(u)	70.5%	1.082	1.437	0.671
	UND(u)	**71.4%**	1.076	1.431	**0.676**
TrustWalker	FWD(u)	40.1%	**0.960**	**1.377**	0.498
	BWD(u)	58.5%	0.970	1.425	0.613
	UND(u)	**66.8%**	0.975	1.394	**0.660**

To summarize the results for cold-start users, the coverage using $UND(u)$ increases to a much larger extent than all-users cases. Consequently, utilizing $UND(u)$ will be more advantageous for

cold-start users with respect to the coverage. On the other hand, the accuracy results do not show much difference from the cases using $FWD(u)$ by using $BWD(u)$ or $UND(u)$. Since it is very difficult to predict the interests for cold-start users, increasing coverage for cold-start users has been an important issue in the recommendation community. We believe that utilizing $UND(u)$ would definitely be a viable alternative.

Overall, we conclude that the ratings prediction using $UND(u)$ can induce increased coverage while maintaining similar accuracy when compared to existing methods, both for the entire users and for the cold-start users. In particular, the coverage increases by a large amount for the cold-start users, suggesting that our approach can become a promising solution to deal with cold-start users.

5. Trust-Aware Imputation

In this chapter, we claim that the reasoning in the backward direction as well as the forward direction should also be valuable in trust-based recommendation methods. That is, a trustee would have interests in those items which her trustors are interested in because similarity relationship is symmetric, driven by both forward and backward directions of edges.

When our imputation method finds the reliable neighbors of the active user, in addition to considering the reachable users through both directions of edges, it also considers the distance between the active user and her reliable neighbors. In this chapter, we denotes the distance threshold as a parameter δ, and we only consider the users who are reachable through forward and backward direction of edges within the distance threshold during the imputation process.

5.1 Imputation of Rating Matrix through Trust Network

In order to predict the ratings by the probabilistic matrix factorization, we represent the ratings given by users to items as a rating matrix $R = (r_{u,i})_{M \times N}$, as those previous works do. The subscripts *M* and *N* mean the numbers of users and items, respectively. The element r_{ui} denotes the rating given by user *u* to item *i*. In the real world, the rating matrices of most domains are very sparse [Ado05][Sar00][Bal97][Paz99][Hua04]. For example, 50% of users only give less than 5 ratings in the Epinions and Flixter datasets [Jam10].

The reachable users from user *u* through both forward links and backward links within distance limitation δ are denoted as $BID_\delta(u)$. On the other side, we define the reachable users from user *u* through only forward links as $FWD_\delta(u)$. In Figure 1.2, if we set $\delta = 2$, we can obtain $FWD_2(U_1) = \{U_4, U_5, U_8\}$ and $BID_2(U_1) = \{U_2, U_3, U_4, U_5, U_6, U_7, U_8, U_9\}$. To impute a rating to target item *i* for active user *u*, the method finds $BID_\delta(u)$ through the associated trust network and uses their ratings of item *i* to estimate the impute rating r'_{ui} as follows:

$$r'_{u,i} = \bar{u} + \frac{\sum_{v \in \{v \mid r_{v,i} \neq 0, v \in BID_\delta(u)\}} w_{u,v} (r_{v,i} - \bar{v})}{\sum_{v \in \{v \mid r_{v,i} \neq 0, v \in BID_\delta(u)\}} w_{u,v}} \quad (5.1)$$

where r_{vi} denotes the rating of user *v* on item *i*, *u* and *v* represent the average rating of user *u* and *v* respectively. As you can see in Equation 5.1, the proposed imputation method considers not all of

the users in $BID_\delta(u)$ but those users who rated the item *i* in $BID_\delta(u)$. Weight $w_{u,v}$ represents the weight which describe the influence of distance between two users *u* and *v* because it likely for two users to have different preference according to the distance between them. We can set it to small values when the distance of two users is greater or simply set it to a constant regardless of the distance.

We analyze the Epinions dataset [Mas07] to confirm the reliable neighbors of each user found through both forward and backward directions of edges also have similar tastes with that user and compare them to the reliable neighbors found through only forward direction. Table 5.1 shows the analysis result, including the similarity between users and their *FWD* and the similarity between users and their *BID*. The similarity scores are computed by Pearson's correlation coefficient, and the distance threshold was 6. Also, when averaging the similarities between each user and her reliable neighbors, we ignore the similarity values that are equal or smaller than 0. Table 5.1 shows the similarity between each user and her *FWD* and the similarity between each user and her *BID* are quite close on average, which can prove our assumption that users also have similar preferences with their trustors.

Table 5.1. Similarity between each user and their BID/FWD

Neighbors	Similarity
$BID_6(\cdot)$	0.694
$FWD_6(\cdot)$	0.691

The proposed method estimates the unrated rating of the target item by aggregating the ratings given by the reliable neighbors; thus, the imputed rating of item that is evaluated by few reliable neighbors would be imprecise. An example of this problem is demonstrated in Figure 5.1: circles represent users and edges between circles represent trust relationships. The table below each user denotes ratings given by that user to items, items without ratings mean those items have not been rated yet by the user. If we want to estimate for the items that are not evaluated by U_1, the ratings given to those items by $BID_2(U_1) = \{U_2, U_3, U_4, U_5, U_6, U_7, U_8, U_9\}$ will be considered when the distance threshold is 2. In the case of item I_2, all the 8 reliable neighbors evaluated it and we can impute rating to I_2 for U_1 by aggregating the opinions of those 8 reliable neighbors. However, in the case of item I_5 which has been rated by only user U_3, we can only use the opinions of U_3 to do imputation, which is not reliable relatively. In reality, not so many items have been rated by more than 8 users; on the other hand, many items just like I_5 have been rated by only a small number of

users. For instance, in the Epinions dataset, 56% of items only rated by one user. In terms of the example in Figure 5.1, although all ratings can be imputed for $U1$ by using Equation 1, ratings of items like I_5 which were rated only by a small number of reliable neighbors cannot be regarded as precise imputation.

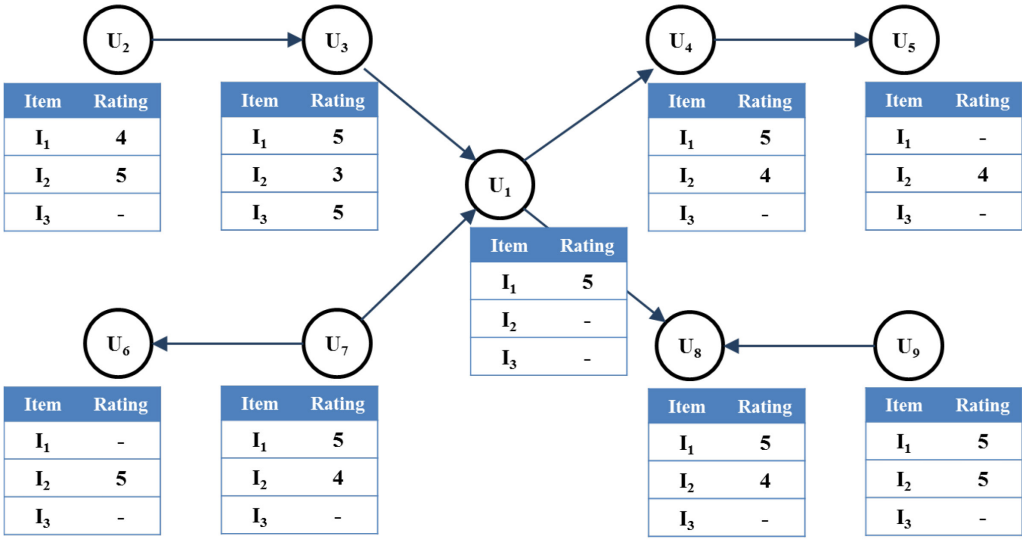


Figure 5.1. An example of imputing U_1's unrated ratings using neighbors' ratings is shown. Item I_5 will be not imputed due to the lack of ratings among the neighbors.

In order to solve this problem, we discard those ratings of items which are only rated by a small number of reliable neighbors for active user u during the imputation process, and only estimate the rest of items, called the *candidate item set* of user u, $C_\theta(u)$. We propose two variations for the choice of a part of the unrated items as the candidate item set for each user. The first one is that the candidate item set includes the item where the number of reliable neighbors who evaluate that item is upper than a predefined the *candidate threshold* θ. For example, if the threshold is 4, the ratings of I_2, I_3 and I_4 for active user U_1 will be included in $C_{\theta=4}(U_1)$ and I_5 will be ignored for U_1 in Figure 5.1.

The other one is that we sort the items for each active user based on the number of her reliable neighbors who evaluate each item in descending order; and then, we include top θ percent of items in the candidate set. In Figure 5.1, if θ = 50%, only ratings of I_2 and I_3 will be included in $C_{\theta=50\%}(U_1)$. We conduct experiments about those variations, and the result shows the later one was a little better. However, due to space limitation, we omit the experimental results of comparison of two variations in this chapter. In addition, the amount of estimated ratings has a strong impact on

the convergence time of the whole recommendation process based on matrix factorization. Thus, the control of imputation percentage is very important in practical application. For that reason, we apply the later variation of conducting the candidate sets to our final imputation method in this chapter, which can be defined as:

$$r'_{u,i\epsilon\{i\,|i\epsilon C_\theta(u)\}} = \bar{u} + \frac{\sum_{v\epsilon\{v\,|r_{v,i}\neq 0, v\epsilon BID_\delta(u)\}} w_{u,v}\cdot (r_{v,i}-\bar{v})}{\sum_{v\epsilon\{v\,|r_{v,i}\neq 0, v\epsilon BID_\delta(u)\}} w_{u,v}} \tag{5.2}$$

5.2 Rating Prediction

In this chapter, we employ the probabilistic matrix factorization [Sal08] to learn the latent characteristics of users and items and use them to predict the unknown ratings. Considering a rating matrix *R*, where there are *M* users and *N* items. The element $r_{u,i}$ denotes the rating of user *u* on item *i*. Actually, $r_{u,i}$ can be any real number, but often ratings are integers from 1 to r_{max}. In this chapter, we map the ratings $r_{u,i}$ to the interval [0, 1] by linearly scaling function $f(x) = (x - 1)/(r_{max} - 1)$ without loss of generality. Let $U \in R^{K\times M}$ and $V \in R^{K\times N}$ represent latent user and item feature matrices; U_u and V_i represent *K*-dimensional user and item latent feature vectors of user *u* and item *i*, respectively. In [Sal08], the conditional probability of the observed rating is defined as:

$$p(R|U,V,\sigma_R^2) = \prod_{u=1}^{M}\prod_{i=1}^{N}[\mathcal{N}(r_{ui}|g(U_u^T V_i),\sigma_R^2)]^{I_{ui}^R} \tag{5.3}$$

where $\mathcal{N}(x|\mu,\sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and variance σ^2, and I_{ui}^R is the indicator function that is equal to 1 if user *u* rated item *i* and equal to 0 otherwise. The function *g*(*x*) is the logistic function $g(x) = 1/(1+e^{-x})$, which bound the range of $U_u^T V_i$ within the range [0, 1]. The zero-mean spherical Gaussian priors are also assumed for user and item feature vectors:

$$p(U|\sigma_U^2) = \prod_{u=1}^{M}\mathcal{N}(U_u|0,\sigma_U^2\mathbf{I}),\ p(V|\sigma_V^2) = \prod_{i=1}^{N}\mathcal{N}(V_i|0,\sigma_V^2\mathbf{I}) \tag{5.4}$$

Hence, through a Bayesian inference, we can obtain the posterior probability of the latent variables *U* and *V* as follows:

$$p(U,V|R,\sigma_R^2,\sigma_U^2,\sigma_V^2) \propto p(R|U,V,\sigma_R^2)p(U|\sigma_U^2)p(V|\sigma_V^2) = \prod_{u=1}^{M}\prod_{i=1}^{N}[\mathcal{N}(R_{ui}|g(U_u^T V_i),\sigma_R^2)]^{I_{ui}^R} \times \prod_{u=1}^{M}\mathcal{N}(U_u|0,\sigma_U^2\mathbf{I}) \times \prod_{i=1}^{N}\mathcal{N}(V_i|0,\sigma_V^2\mathbf{I}) \tag{5.5}$$

The log of the posterior probability over the user and item features is given by

$$\ln p(U,V|R,\sigma_R^2,\sigma_U^2,\sigma_V^2) = -\frac{1}{2\sigma_R^2}\sum_{u=1}^{M}\sum_{i=1}^{N} I_{ui}^R \left(R_{ui} - g(U_u^T V_i)\right)^2 - \frac{1}{2\sigma_U^2}\sum_{u=1}^{M} U_u^T U_u -$$

$$\frac{1}{2\sigma_V^2}\sum_{i=1}^{N} V_i^T V_i - \frac{1}{2}\left(\left(\sum_{u=1}^{M}\sum_{i=1}^{N} I_{ui}^R\right)\ln\sigma_R^2 + (M \times K)\ln\sigma_U^2 + (N \times K)\ln\sigma_V^2\right) + C \quad (5.6)$$

where *C* is a constant that does not depend on the parameters. Keeping the parameters (observation noise variance and prior variances) fixed, maximizing the log-posterior over two latent features is equivalent to minimizing the following sum-of-squared-errors objective function with quadratic regularization terms:

$$\mathcal{L}(R,U,V) = \frac{1}{2}\sum_{u=1}^{M}\sum_{i=1}^{N} I_{ui}^R \left(R_{ui} - g(U_u^T V_i)\right)^2 + \frac{\lambda_U}{2}\sum_{u=1}^{M}\|U_u\|_F^2 + \frac{\lambda_V}{2}\sum_{i=1}^{N}\|V_i\|_F^2 \quad (5.7)$$

where $\lambda_U = \sigma_R^2/\sigma_U^2$, $\lambda_V = \sigma_R^2/\sigma_V^2$ and $\|\cdot\|_F^2$ denotes the Frobenius norm.

A local minimum of the objective function can be found by performing a gradient descent search on U_u and V_i for all user *u* and all item *i*.

$$\frac{\partial \mathcal{L}}{\partial U_u} = \sum_{i=1}^{N} I_{ui}^R \, g'(U_u^T V_i)(g(U_u^T V_i) - r_{ui})V_i + \lambda_U U_u \quad (5.8)$$

$$\frac{\partial \mathcal{L}}{\partial V_i} = \sum_{u=1}^{M} I_{ui}^R \, g'(U_u^T V_i)(g(U_u^T V_i) - r_{ui})U_u + \lambda_V V_i \quad (5.9)$$

where $g'(x)$ is the derivative of logistic function, $g'(x) = e^{-x}/(1+e^{-x})^2$. In order to reduce the model complexity, in all our experiments we set $\lambda_U = \lambda_V$.

When estimates of *U* and *V* are found, we can predict rating of user *u* on item *i* as follows:

$$\hat{r}_{ui} = g(U_u^T V_i) \quad (5.10)$$

5.3. EXPERIMENTS

In this section, we perform various experiments to clearly demonstrate our contributions. Our experiments show that our trust-based imputation method has better recommendation accuracy than previous works, especially for cold start users.

5.3.1 Experimental Setup

We use three real-world data sets that are used by previous researches [Mas07, Jil12] for the experiments. Two of those data sets are collected from Epinions, and the last one is collected from Ciao. For convenience, we name those data sets as Epinons1, Epinions2, and Ciao respectively.

Table 2 shows the statistics of each data set. The users of Epinions1 data set are more than those of other data sets, and the items and ratings of Epinons2 data set are more than those of the other data sets. The number of trust relationships in the Epinions1 data set is the highest among those of data sets. The size of Ciao data set is smaller than those of the other two data sets. In Table 5.2, we define the cold-start users as those users who have evaluated less than five items like the existing researches [Mas07, Jam09, Jam10]. Although it is widely known that the most of users are generally the cold-start users in many domains, there are only few cold start users in Epinions2 and Ciao data sets. This is because that the previous researches which collected those data sets may not have included the cold start users. The sparsity means the ratio of the number of nonzero elements to the rating matrix, and table2 shows that all of the data sets are very sparse.

Table 5.2. Statistics of data sets

Statistics	**Data set**		
	Epinions1	**Epinions2**	**Ciao**
Users	49,289	22,166	7,375
Items	139,738	296,277	106,797
Ratings	664,824	916,085	282,619
Trusts	487,002	355,813	111,781
Cold Users	16,910	21	6
Sparsity	0.0001	0.0004	0.0001

Our imputation method needs to find the reliable neighbors of each user for estimating her unrated ratings. The reliable neighbors of an active user are those who are reachable from her within the distance threshold δ in the trust network. Also, our method determines the candidate item set as the candidate threshold θ. In the proposed recommendation methods, the two parameters which the distance threshold δ and the candidate threshold θ not only determine the accuracy of imputed ratings but also influence that of prediction results. For this reason, we vary those two parameters and find the best settings of them by comparing the accuracy among different parameter settings.

In order to measure the prediction accuracy, we use *mean absolute error (MAE)* and *root mean square error (RMSE)* as evaluation metrics in our experiments. MAE is a measure that calculates the average of the differences between the predicted ratings and the actual ratings while RMSE is a measure that puts more emphasis on big errors. We perform 5-fold cross validation in our experiments. In each fold we use 80% of rating data as the training set and the remaining 20% as the test set. We use the same setting for all other experiments.

In order to impute ratings for an active user, the proposed method selects users who are reachable from the active user through both the forward and backward directions of edges as the reliable neighbors. Unlike our method, the existing trust-based recommendation methods only estimate the ratings for an active user by considering those users who are reachable from the active user through the forward direction. In order to check the effectiveness of finding the reliable neighbors through not only the forward direction but also the backward direction of edges, we perform the experiment that compares two set of reliable neighbors: the one includes the users found by bi-directions and the other by forward directions. In addition, we perform the same experiment for only the cold start users because the rating prediction for them may be less accurate than that for other users. This is because the number of reliable neighbors of the cold start users may be very low when following only forward directions.

After finding the best combination of two parameters, we conduct the experiment to show the use of the trust network is helpful for precise imputation. In order to show that result of imputation is precise, we check the accuracy of prediction using the imputed ratings. For the experiment, we introduce the simple imputation method, called *SimVote*, which does not consider the trust network, and compare it with our imputation method. SimVote fills the unrated ratings for an active user with average ratings of her based on the default voting [Bre98] approach which is a simple one among the existing imputation methods. To impute similar number of the unrated ratings to our imputation method, we randomly select θ percent of the unrated ratings and impute only those ratings by SimVote. For predicting the ratings, we apply the probabilistic matrix factorization to the rating matrix that is imputed by SimVote.

Finally, we conduct experiments for comparing the recommendation method using the proposed imputation method with the existing ones. In these experiments, we choose four existing recommendation methods: the user-based recommendation method [Res94], AutAI [Ren12], PMF [Sal08], and SocialMF [Jam10]. The reason why we select the user-based recommendation method is that it is the well-known and simple collaborative filtering method. Also, we select the AutAI because it is one of the latest imputation based recommendation. For AutAI, we set the parameter λ to 0.4, which describes the ratio between the user aspect and item aspect imputation. Since the proposed recommendation method exploits the probabilistic matrix factorization for predicting the ratings, we consider PMF as the comparison method. Moreover, we choose the SocialMF since it also employs the matrix factorization and overcomes the data sparsity problem by looking up the trust network. We set the parameter λT to 0.5 for SocialMF, which determines the influence of the social network in the rating prediction. In the proposed imputation method, we set w_{uv} that

describes the weight of distance between user u and v to same value 1, which shows best accuracy among the various values. Due to the page limit, we omit the experiments for determining w_{uv} in this chapter. We perform 5-dimensional matrix factorization in PMF, SocialMF and our proposed method, and we set the parameter λU and λV to 0.01 for those methods. Besides performing experiments on all users, we also perform the same experiment for only the cold start users because the rating prediction for them may less accurate than that for other users, which is a serious problem of existing recommender systems.

5.3.2 Experimental Results

The prediction accuracy using the proposed imputation method with different parameter settings has been showed in Tables 5.3, 5.4, and 5.5. One parameter is the distance threshold δ that determines the reliable neighbors, and the other is the candidate threshold θ that represents the ratio of the unrated ratings that will be estimated to whole unrated ratings. Tables 5.3, 5.4, and 5.5 show the experimental results on the Epinions1, Epinions2, and Ciao data set respectively. In those tables, the dark cells represent the best case of two parameters in terms of RMSE and MAE.

In Tables 5.3 and 5.5, we can observe that our method has the lowest RMSE when the candidate threshold θ is 20%, while θ is 30% in the case of MAE. However, setting the candidate threshold θ as 10% achieves the lowest RMSE while setting θ as 20% does the lowest MAE in Table 5.4, which is different from other tables. This is because that the Epinions2 data set has more ratings compared to the Epinions1 and Ciao data set. In most cases, our method produces the lowest RMSE or MAE when the distance threshold δ is 3 or 4 in Tables 5.3, 5.4, and 5.5. MAE in Table 5.3 shows different tendency to the other results, and it has lowest value when δ is 6.

Those results show setting the distance threshold as 3 or 4 and the candidate threshold as 20% seems to produce the best accuracy in most of data sets regardless of the measurement. It seems to be reasonable because Mole Trust [Mas07] also claims that the users who are reachable from the active user within the distance 4 are appropriate to accurate prediction. Besides, the best candidate threshold is not high because it seems to be that more imputed ratings cannot guarantee the better accuracy in the prediction using the matrix factorization technique.

Table 5.3. Effect of distance threshold δ and candidate threshold θ in Epinions1 data set

Measure	Candidate Threshold(θ)	Distance Threshold(δ)					
		1	2	3	4	5	6
RMSE	10%	1.1089	1.0941	1.0959	1.0966	1.0968	1.0968
	20%	1.1073	1.0886	1.0876	1.0877	1.0878	1.0879
	30%	1.1072	1.0934	1.0899	1.0891	1.0892	1.0893
	40%	1.1062	1.0962	1.0935	1.0953	1.0950	1.0950
	50%	1.1051	1.1003	1.1001	1.1013	1.1015	1.1023
MAE	10%	0.8666	0.8466	0.8401	0.8392	0.8395	0.8395
	20%	0.8626	0.8370	0.8310	0.8295	0.8292	0.8291
	30%	0.8616	0.8433	0.8271	0.8232	0.8227	0.8226
	40%	0.8626	0.8437	0.8296	0.8279	0.8281	0.8280
	50%	0.8637	0.8450	0.8310	0.8294	0.8284	0.8284

Table 5.4. Effect of distance threshold δ and candidate threshold θ Epinions2 data set

Measure	Candidate Threshold(θ)	Distance Threshold(δ)					
		1	2	3	4	5	6
RMSE	10%	1.1041	1.0900	1.0843	1.0851	1.0855	1.0860
	20%	1.1030	1.0923	1.0865	1.0907	1.0913	1.0916
	30%	1.1013	1.0944	1.0942	1.1018	1.1025	1.1029
	40%	1.0994	1.0969	1.1102	1.1204	1.1210	1.1249
	50%	1.0977	1.1023	1.1519	1.1886	1.1984	1.1995
MAE	10%	0.8631	0.8404	0.8293	0.8281	0.8282	0.8294
	20%	0.8618	0.8460	0.8285	0.8281	0.8287	0.8300
	30%	0.8605	0.8465	0.8321	0.8329	0.8334	0.8353
	40%	0.8590	0.8469	0.8395	0.8421	0.8420	0.8456
	50%	0.8576	0.8501	0.8602	0.8798	0.8794	0.8752

Table 5.6 shows the results of different sets of the reliable neighbors with the various distance threshold δ and directions of edges in terms of RMSE. In this experiment, we use three data sets, and set the candidate threshold θ as the best value which from Tables 5.3, 5.4, and 5.5. In Table 5.6, $FWD_\delta(u)$ represents the set of reliable neighbors who are reachable from the active user through the forward direction within the distance δ while $BWD_\delta(u)$ includes the reliable neighbors reachable through the bi-directions of edges. The sets of reliable neighbors found through the bi-directions produce higher accuracy in all data sets. Especially, in the Epinions2 data set, using the set of reliable neighbors found through the bi-directions shows much better accuracy, while the

accuracy of using this set is not significantly higher in other two data sets.

Table 5.5. Effect of distance threshold δ and candidate threshold θ in Ciao data set

Measure	Candidate Threshold(θ)	Distance Threshold(δ)					
		1	2	3	4	5	6
RMSE	10%	1.0088	0.9899	0.9863	0.9881	0.9893	0.9899
	20%	1.0052	0.9864	0.9813	0.9811	0.9812	0.9816
	30%	1.0045	0.9867	0.9816	0.9826	0.9827	0.9831
	40%	1.0039	0.9895	0.9830	0.9834	0.9880	0.9934
	50%	1.0036	0.9907	0.9884	0.9884	0.9980	0.9982
MAE	10%	0.7843	0.7571	0.7505	0.7521	0.7556	0.7563
	20%	0.7781	0.7485	0.7379	0.7377	0.7380	0.7407
	30%	0.7759	0.7476	0.7339	0.7348	0.7348	0.7355
	40%	0.7811	0.7599	0.7516	0.7493	0.7484	0.7497
	50%	0.7743	0.7465	0.7325	0.7338	0.7521	0.7525

Table 5.6. Effect of edges' direction and distance threshold δ for whole users

Data	Edges' Direction	Distance Threshold (δ)					
		1	2	3	4	5	6
Epinions1 (θ=20%)	BID(u)	1.1073	1.0886	**1.0876**	1.0877	1.0878	1.0879
	FWD(u)	1.1100	1.0968	1.0895	**1.0877**	1.0880	1.0880
Epinions2 (θ=10%)	BID(u)	1.1041	1.0900	**1.0843**	1.0851	1.0855	1.0860
	FWD(u)	1.1059	1.1025	1.0929	**1.0905**	1.0912	1.0915
Ciao (θ=20%)	BID(u)	1.0052	0.9864	0.9813	**0.9811**	0.9812	0.9816
	FWD(u)	1.0770	0.9919	0.9843	**0.9822**	0.9822	0.9825

Table 5.7 shows similar results as Table 5.6 for the cold start users. As shown in Table 5.2, Epinions2 and Ciao data set barely include the cold start users, so we perform the experiment on only the Epinions1 data set. In Table 5.7, the RMSE of all cases are worse than that of Table 5.6 because the cold start users have fewer trust relationships, which are not sufficient for accurate prediction. Also, the accuracy employing bi-directions of edges is higher than that of only using forward directions. Especially, the difference between two cases in terms of RMSE is larger than that in Table 5.6. The cold start users are likely to have fewer trust relationships than the other users, thus using backward directions of the edges additionally that makes possible to find more number

of reliable neighbors will improve the accuracy much higher.

Table 5.7. Effect of edges' direction and distance threshold δ for cold start users

Data	Edge's Direction	Distance Threshold (δ)					
		1	2	3	4	5	6
Epinions1 (θ=20%)	BOTH(u)	1.1995	1.1698	1.1492	1.1427	**1.1415**	1.1418
	FWD(u)	1.2027	1.1911	1.1737	1.1581	**1.1539**	1.1541

Figure 5.2 shows the accuracy of prediction using two imputation methods: the proposed imputation method and SimVote that randomly selects the unrated ratings and imputes them by default voting approach [3]. In Figure 5.2, x-axis and y-axis represent the candidate threshold θ and the values of MAE or RMSE, respectively. Each subfigure corresponds to each data set. As you can see in Figure 5.2, using the proposed imputation method is more accurate than using SimVote regardless of the value of θ and the measurement. The reason is that the proposed imputation method only imputes those unrated ratings that would be estimated precisely whereas SimVote does not consider this. Therefore, the accuracy of prediction using SimVote is certainly lower. In Figure 5.2, the interesting point is that the more imputed ratings, the worse prediction accuracy. This is because that the methods via matrix factorization predict ratings accurately when the imputation method only imputes the ratings that would be estimated precisely even if there is only small number of imputed ratings. In other words, the more imprecise imputed ratings make the worse accuracy of prediction.

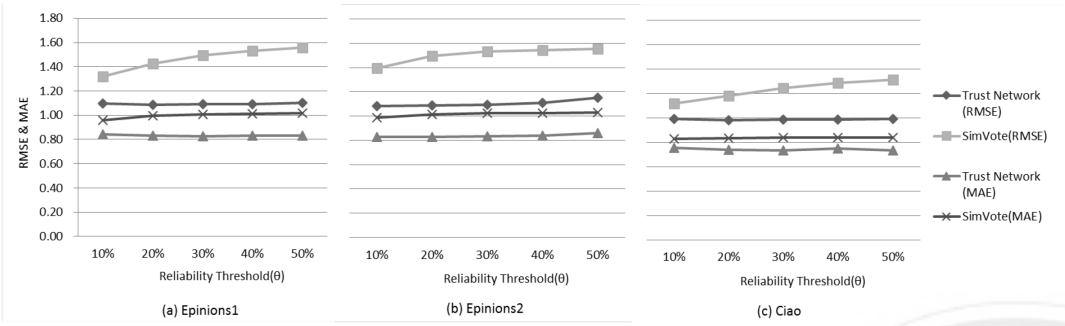


Figure 5.2. Comparison proposed imputation method with SimVote.

Table 5.8 shows the accuracy of the proposed imputation-based recommendation method and the existing recommendation methods. The proposed method is more accurate than others especially on Ciao data set. Moreover, the proposed method and SocialMF overcome PMF, which means the use

of trust network is advantageous to the recommendation accuracy. Besides, the proposed method is more accurate than SocialMF due to two reasons. The one reason is that the proposed method considers those users who are reachable through the backward direction, and the other is that it only considers those users who within the distance threshold which can be seen as more reliable. AutAI employing the imputation approach defeats other existing recommendation methods although it does not use matrix factorization. It seems that all the three data sets are very sparse, so those methods which employ imputation approaches may produce higher accuracy than others.

Table 5.8. Accuracy of recommendation methods for whole users

Dataset	Method	RMSE	MAE
Epinions1	User-based	1.185	0.894
	AutAI	1.097	0.838
	PMF	1.118	0.880
	SocialMF	1.115	0.879
	Ours	**1.088**	**0.823**
Epinions2	User-based	1.136	0.837
	AutAI	1.090	0.835
	PMF	1.109	0.868
	SocialMF	1.106	0.870
	Ours	**1.084**	**0.828**
Ciao	User-based	1.179	0.849
	AutAI	1.006	0.736
	PMF	1.033	0.810
	SocialMF	1.030	0.808
	Ours	**0.981**	**0.734**

Table 5.9 shows the accuracy of each method for the cold start users. In Table 5.9, all methods show worse accuracy than Table 5.8 because the cold start users have left less information. In this table, like Table 5.8, the proposed method has higher accuracy than other existing recommendation methods. In addition, the differences between the proposed method and those existing methods in terms of RMSE are much bigger in Table 5.9 compared to those in Table 5.8. AutAI overcomes PMF and SocialMF in Table 5.8 whereas it does not in Table 5.9. This is because that the imputation for the cold start users cannot impute reliable ratings due to the lack of information for the cold start users. Unlike AutAI, the proposed method can predict the accurate ratings because it refers to additional information, the trust network, to impute the unrated ratings for the cold start users.

Table 5.9. Accuracy of recommendation methods for cold start users (Epinions1)

Method	RMSE	MAE
User-based	1.350	0.993
AutAI	1.288	0.934
PMF	1.210	0.962
SocialMF	1.209	0.968
Ours	**1.142**	**0.888**

Through the extensive experiments, we found the most appropriate parameters for finding the reliable neighbors in the proposed imputation method, and the proposed imputation method is helpful improving the accuracy of rating prediction. In addition, the recommendation method that employs our imputation method through the trust network shows better accuracy than the method using the simple imputation method without considering the trust network. At last, we showed that the proposed method overcomes the existing recommendation methods in terms of RMSE and MAE, especially for the cold start users.

6. Using Category Experts

6.1. Category Experts Methods

This section proposes four recommendation methods using category experts. The *category expert* is a user who is considered to understand well the overall items in a specific category. In this chapter, the decision of whether or not a user understands well the item is based on whether or not the user has evaluated the item. The reasoning behind this is, in order to evaluate an item, the user needs to know the item well. Therefore, a user who has evaluated many items in a specific category can be considered as *knowledgeable about the category*, thereby defined as a *category expert*. We formally define the category expert as follows:

Definition 1. For category c, we define the category experts E_c such that:

$$|I_u| \leq |I_v| \ (\forall v \in E_c)$$

where $|I_u|$ indicates the number of items that user u has evaluated. In order to determine the category experts, we compute the number of items in each category evaluated by each user. It is easy to maintain those numbers incrementally because we just need to increment the number whenever a user evaluates an item. For this reason, we can reduce dramatically the running time of NBMs by decreasing the effort for finding neighbors (i.e., category experts).

CE method: In this chapter, we denote the recommendation method that uses category experts as a CE method. The CE method predicts the rating with which the active user would assign to the target item, based on the ratings given by the category experts. The intuition behind this is that a user trusts the opinions of experts even if she has somewhat different preference with the category experts. The predicted rating for user u on item i included in category c is denoted as $p_{u,i,c}$.

$$p_{u,i,c} = \bar{r}_{u,c} + \frac{1}{k}\sum_{v \in E_c}(r_{v,i} - \bar{r}_{v,c}) \tag{6.1}$$

$\bar{r}_{u,c}$ is an average rating assigned by user u to the items in category c. E_c is an expert group of category c and consists of the top-k users who assigned the most ratings on the category. $r_{v,i}$ is the rating of user v in expert group E_c assigned to item i. Most items belong to a single category and their expected ratings can be calculated by Equation (6.1). When dealing with multiple categories, we consider the item ratings by all experts from their belonging categories. In this approach, the rating $p_{u,i}$ on item *i* for user u can be predicted in the following.

$$p_{u,i} = \frac{1}{|C_i|}\sum_{c \in C_i} p_{u,i,c} \quad (6.2)$$

C_i indicates a set of categories where item i is included, and $|C_i|$ is the number of those categories.

While the CE method needs a short running time, its accuracy is lower than other proposed methods that we will introduce later. This is because, regardless of an active user, the CE method provides almost the same predicted rating for the same item. Specifically, it predicts a rating only based on the category experts who are selected independently of active users.

CES method: The CE method ignores that a user tends to agree on opinions of other users who have similar tastes to the user [Bre98]. Based on this observation, we assume that users eagerly accept the opinions of category experts who have similar tastes, but not from all other experts. We adopt this assumption and extend the CE method (refer to as a CES method). The CES method calculates the similarity between the category experts and the active user to measure how close their interests are. $s_{u,v}$ represents the similarity between user u and v, by the Pearson's correlation coefficient shown in Equation (6.3).

$$s_{u,v} = \frac{\sum_{i \in I}(r_{u,i} - \bar{r}_u)\cdot(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I}(r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_{i \in I}(r_{v,i} - \bar{r}_v)^2}} \quad (6.3)$$

I is the set of items to which both user u and category expert v have assigned ratings. $r_{u,i}$ is the rating assigned by user u on item i, and $\bar{r}_u$ represents the average of all the ratings assigned by user u. The predicted rating $p_{u,i,c}$ of target item i at category c is calculated by Equation (6.4). We note it additionally uses the similarity $s_{u,v}$ between user u and expert v in ratings prediction.

$$p_{u,i,c} = \bar{r}_{u,c} + \frac{\sum_{v \in E_c}(r_{v,i} - \bar{r}_{v,c})\cdot s_{u,v}}{\sum_{v \in E_c} s_{u,v}} \quad (6.4)$$

The rating $p_{u,i,c}$ is only determined by the ratings of category experts at category c, so there are several $p_{u,i,c}$ when item i is included in multiple categories. The CES method aggregates all these predicted ratings $p_{u,i,c}$ to determine the final rating $p_{u,i}$ for user u on item i by applying Equations (6.4) to (6.2).

The CES method shows better accuracy than the CE method because it gives different weights to the opinions of category experts depending on an active user. On the other hand, it needs more execution time than the CE method due to the extra overhead for similarity computations. Even though it computes the similarity of users, it spends much less execution time than the existing UBM. This is because the CES method has to compute *k* similarity values for an active user while UBM does *n* similarity values. We note k and n are the numbers of category experts and all users,

respectively.

CEP method: Some items are belonging to multiple categories. Both the CE and CES methods equally consider the opinions of each category's experts. In the real world, a user may have different levels of interest in each category and is more likely to respect the opinions of the users who have expertise on their favorite category. Based on this intuition, we propose the CEP method that is an extended version of the CE method. In the CEP method, we consider the category experts' opinions differently depending on the active user's interest on each category to enhance the accuracy of rating prediction. The CEP method first defines the category interest as the interest level of a user in a category. The category interest of a user on a category is proportional to the number of items that she has evaluated in the category. The CEP method predicts rating $p_{u,i}$ on item i for user u as Equation (6.5).

$$p_{u,i} = \frac{\sum_{c \in C_i} p_{u,i,c} \cdot f_{u,c}}{\sum_{c \in C_i} f_{u,c}} \tag{6.5}$$

C_i denotes the categories that item i belongs to, and $f_{u,c}$ is the amount of interest that user u has in category c. In the CEP method, $p_{u,i,c}$, which is calculated by Equation (6.1) is applied to Equation (6.5), to predict the final rating.

The accuracy of the CEP method is higher than that of the CE method because it produces more personalized results based on the active user's category interest. In terms of the execution time, the CEP method performs worse than the CE method because it additionally utilizes category interests $f_{u,c}$ (compare the Equations (6.2) and (6.5)). However, we expect that the additional execution time is insignificant because each user's category interests are already computed to select the category experts. For this reason, compared to the CES method, the CEP method performs much faster.

CESP method: The similarities between the active user and the category experts can be applied to CE together with the category interest. In a CESP method, the rating of user u on item i is predicted through Equation (6.4) in the CES method for $p_{u,i,c}$. The calculated $p_{u,i,c}$ is applied to Equation (6.5) to obtain $p_{u,i}$, which is the final predicted rating that is a result of aggregating the predicted ratings $p_{u,i,c}$.

$$p_{u,i} = \frac{1}{\sum_{c \in C} f_{u,c}} \times \sum_{c \in C} \left(\bar{r}_{u,c} + \frac{\sum_{v \in E_C} (r_{v,i} - \bar{r}_{v,c}) \cdot s_{u,v}}{\sum_{v \in E_C} s_{u,v}} \right) \cdot f_{u,c} \tag{6.6}$$

Table 6.1 summarizes the characteristics of four proposed methods. We propose the CE method to relieve the running time problem of UBMs. The CES, CEP, and CESP methods require additional operations on top of the CE method; yet, they are expected to be faster than UBMs because the

computing overhead of category interest and similarity would be much less than the overhead of calculating similarity among all users. The CESP method is the most accurate than all other proposed methods (i.e., the CE, CESP and CEP methods). More details are discussed in Section 6.2.

Table 6.1. CE and its extensions

Method	Similarity	Category interest
CE	X	X
CES	O	X
CEP	X	O
CESP	O	O

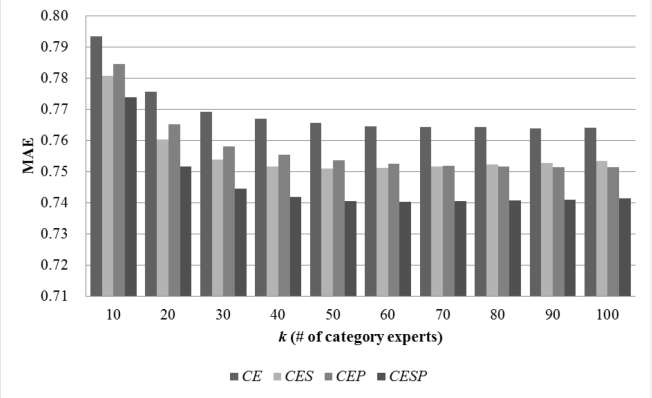
6.2. Evaluation

We used MovieLens and Ciao datasets [Mil03][Tan12] for our evaluation. MovieLens has 100,000 ratings that range from 1 to 5 points for 1,682 movies by 943 users, and every user has rated at least 20 movies. There are 19 movie genres (i.e., one category for one genre), and every movie is classified into at least one genre. Ciao has 26,660 ratings for 1,848 items under 28 categories by 590 users. Every user has rated at least 5 items because we removed the users who have evaluated less than 5 items by following the assumption in [Mas04] to avoid the new (cold-start) user problem [Ado05, Sha11]. The density of MovieLens and Ciao datasets are 0.06 and 0.02, respectively.

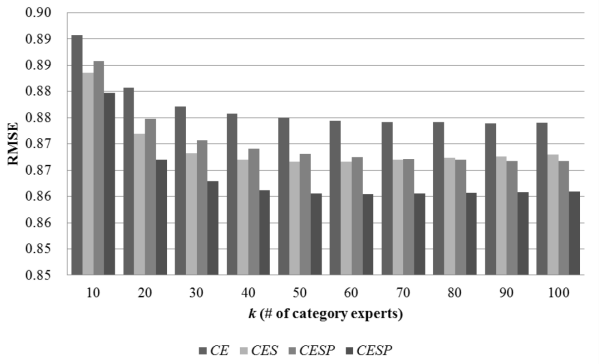
We carried out the experiments with a different number of category experts to measure its effectiveness. The accuracy was measured by dividing the MovieLens dataset into 5 subsets and using cross-validation which is a widely accepted in the recommendation systems area [Ama09, Yun01]. To evaluate the accuracy, we used Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). MAE is a measure that calculates the average error, which is the difference between the predicted ratings and the actual ratings, while RMSE is a measure putting more emphasis on larger errors.

Figure 6.1 shows MAE and RMSE according to different numbers of category experts. The CES and CEP methods show higher accuracy than the CE method. This is due to the fact that considering the similarity between the active user and category experts helps to improve the accuracy of recommendations, which coincides with the results of the existing research [Bre98]. The CESP method shows the highest accuracy. As a result, we can confirm that combining the similarity between the active user and category experts and the category preference of the active user helps

improving the accuracy as well. When the category expert count is 60, the CES, CEP, and CESP methods show the lowest MAE and RMSE values. The CE method shows the lowest values when k is 80. The lowest MAE (RMSE) values of the CE, CES, CEP, and CESP methods are 0.764, 0.751, 0.752, and 0.740 (0.874, 0.867, 0.867, and 0.860), respectively.



a. MAE values for CE, CES, CEP, and CESP.



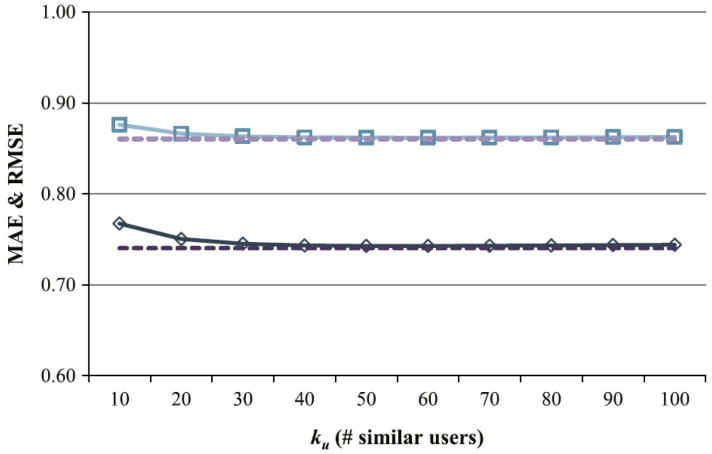
b. RMSE values for CE, CES, CEP, and CESP.

Figure 6.1. Accuracy measures for CE, CSE, CEP, and CESP; a. MAE; b. RMSE.

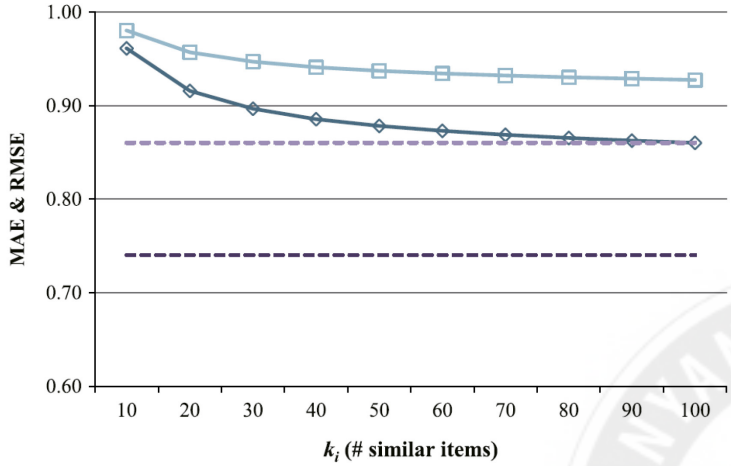
For accuracy, we compared the CE method and its extensions to the following collaborative filtering methods: the user-based method (UBM), the item-based method (IBM), and the k-means clustering based method (CBM). For CBM, we employed k-means clustering that produces different results depending on initial seeds. To avoid the problem caused by its randomness, we performed CBM 10 times and selected the most accurate one as the final result. We tried various parameter values, such as the similar user count (k_u) for UBM, the similar item count (k_i) for IBM, and the cluster count (k_c) for CBM. We evaluated their accuracy according to the parameter values. We

compared their accuracy to the best one of *CESP* with the category expert count of 60.

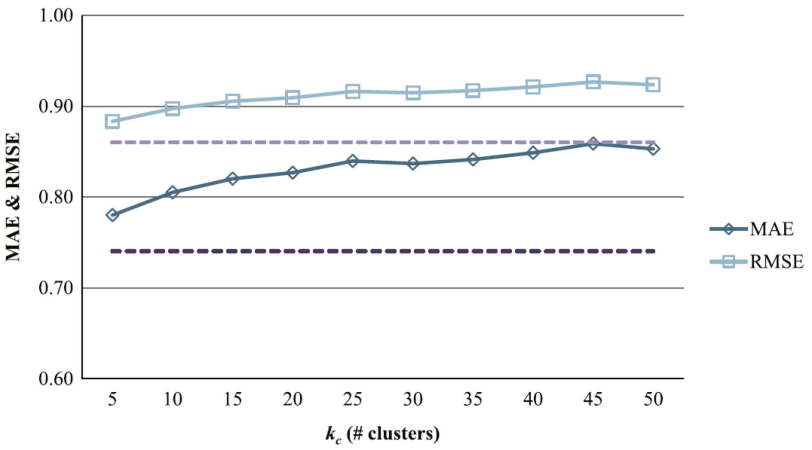
Figure 6.2 presents the accuracy with different parameters. The light-colored dotted line represents RMSE of CESP, and the dark-colored dotted line represents MAE of the CESP method, which are presented for the comparison purpose. Figure 6.2.a shows that UBM has the highest accuracy where k_u is 60 (MAE: 0.743, RMSE: 0.862). The MAE and RMSE values of UBM are higher than those of the CESP method. In addition, we observed that the CE, CES, and CEP methods show lower accuracy than UBM in Figures 6.1 and 6.2.a. In Figure 6.2.b, IBM has the best accuracy with the similar user count of 100 (MAE: 0.860, RMSE: 0.928). The MAE and RMSE values are, however, higher than those of the CESP method. In Figures 6.2.c, CBM with 5 clusters shows the highest accuracy (MAE: 0.780, RMSE: 0.833). As the cluster count increases, the accuracy decreases. CBM shows low accuracy when compared with CESP.



a. User-based method



b. Item-based method



c. Clustering-based method

Figure 6.2. MAE and RMSE values for all methods with different parameters.

We claim that IBM and CBM cannot reflect the latest ratings given by users since these methods rely on the pre-computed values before recommendation requests. To show this limitation, we made new training sets that contains only 90%, 92.5%, 95%, and 97.5% ratings, and then we applied this set to IBM and CBM to build the similarity and clustering model. After that, we used original training sets for predicting ratings in IBM and CBM. The parameters of each method were set as $k_i = 100$ (50) and $k_c = 5$ (5) for Movielens (Ciao).

Table 6.2 shows the MAE and RMSE values of IBM and CBM with new training sets that exclude some ratings. The results show that both IBM and CBM produce higher MAE and RMSE values as they use a less number of ratings for a training set. It means that IBM and CBM are less accurate when they ignore the latest ratings given by users. In IBM with Ciao, the differences of MAE and RMSE values among different training sets are small because Ciao has relatively a small number of ratings considering its numbers of users and items. For this reason, the ratings are insufficient to make accurate results even when no ratings are excluded from the training set. On the other hand, CBM has similar MAE and RMSE values for different training sets in Movielens. This is because Movielens has relatively a more number of ratings, so CBM can build similar clustering models even when training sets exclude some ratings.

We listed the lowest MAE and RMSE values for each method using the Movielens and Ciao datasets in Table 6.3. In addition, we compared them with the values of excluding 5% ratings from the training set for IBM and CBM. The parameters of each method were set as k=60 (100), k_u=60 (100), k_i=100 (50), and k_c=5 (5) for Movielens (Ciao). In this evaluation, we adopted 5-fold cross validation, so the MAE and RMSE values for different folds (pairs of training and test sets) could

be different. In order to see the degree of difference, we show standard deviation in the parenthesis. In the results, the standard deviation values are very small, so the MAE and RMSE values for different folds are similar in each method.

Table 6.2. The MAE and RMSE values for IBM and CBM using the training sets excluding some ratings

Methods	Ratio of ratings (%)	Movielens		Ciao	
		MAE	RMSE	MAE	RMSE
IBM	100	0.810	0.900	0.831	0.911
	97.5	0.811	0.901	0.835	0.913
	95	0.813	0.902	0.836	0.914
	92.5	0.814	0.902	0.837	0.914
	90	0.816	0.903	0.837	0.914
CBM	100	0.784	0.885	0.868	0.932
	97.5	0.783	0.885	0.870	0.933
	95	0.784	0.885	0.871	0.933
	92.5	0.785	0.885	0.874	0.934
	90	0.786	0.887	0.877	0.937

For Movielens, CESP shows higher accuracy than IBM and CBM. The MAE value of CESP is up to 9% lower than that of IBM and 5% lower than that of CBM. CESP even achieves better accuracy than UBM. IBM using all ratings shows better accuracy than that using 95% ratings. In CBM, the accuracy does not change when using 95% ratings are used, since the clustering model does not change significantly. The MAE and RMSE values of all the methods with Ciao are higher than those with Movielens because Ciao is sparser than Movielens (i.e., the densities of Ciao and Movielens are 0.02 and 0.06, respectively). IBM is more accurate than CESP while IBM without 5% ratings is less accurate. The comparison between UBM and CESP is similar to that between IBM and CESP. CBM provides the lowest accuracy and the lowest coverage among all the methods. The accuracies of UBM and IBM are similar to each other for Ciao because the average number of items evaluated by each user (about 36.2) is not significantly different than that of users who have evaluated each item (about 22.7).

In order to show that the CESP method and the existing methods provide statistically different results, we employed the Wilcoxon signed rank test [37] by comparing the ratings predicted by the CESP and existing methods (i.e., UBM, IBM, and CBM). Table 6.4 shows the results of the

Wilcoxon tests for Movielens and Ciao datasets. We observe that all significance values are 0; therefore, we reject the null hypothesis that there is no difference in ratings between the CESP and existing methods. In other words, the CESP method predicts ratings statistically different from the ratings predicted by the existing methods. Observing Tables 6.3 and 6.4, we conclude that the CESP method outperforms UBM, IBM, and CBM on both Movielens and Ciao datasets, which is statistically significant.

Table 6.3. Best MAE and RMSE values for all the methods

Recommendation Methods	**Movielens**		**Ciao**	
	MAE (std. dev.)	**RMSE (std. dev.)**	**MAE (std. dev.)**	**RMSE (std. dev.)**
CESP	**0.740** (0.007)	**0.860** (0.004)	0.836 (0.004)	0.914 (0.002)
UBM	0.743 (0.004)	0.862 (0.003)	0.838 (0.002)	0.915 (0.001)
IBM	0.810 (0.004)	0.900 (0.002)	**0.831** (0.005)	**0.911** (0.003)
IBM (95% ratings)	0.863 (0.004)	0.929 (0.003)	0.838 (0.004)	0.915 (0.002)
CBM	0.780 (0.006)	0.883 (0.003)	0.876 (0.001)	0.936 (0.000)
CBM (95% ratings)	0.780 (0.006)	0.883 (0.003)	0.892 (0.010)	0.944 (0.005)

Table 6.4. The results of Wilcoxon ranks tests for comparing the CESP and existing methods

Data sets	**Statistics**	**Pair of methods compared**		
		CESP-UBM	**CESP-IBM**	**CESP-CBM**
Movielens	Z	-57.189	-91.561	-60.014
	Asymp. Sig. (2-tailed)	.000	.000	.000
Ciao	Z	-5.885	-10.719	-5.370
	Asymp. Sig. (2-tailed)	.000	.000	.000

We illustrate the improvements in terms of execution time. We ran the 64 bit Windows Server 2008 with a 12GB RAM and a 3.50GHz Intel Core-i7 processor and measured the execution time

for predicting 100,000 ratings. The parameters k, k_u, k_i, and k_c from Figures 3 and 4 denote the numbers of category experts, similar users, similar items, and clusters required in each method.

Figure 6.3 shows that the more category experts there are, the longer the execution time is. The left among the two y-axes indicates the execution time of CE and CEP while the right one shows the time of CES and CESP. We can see that CE and CEP spend relatively less execution time since they are not required to calculate the similarities between the active user and category experts. In Figure 1, there is almost no change in accuracy when the numbers of category experts is more than 40; Figure 3 shows that the execution time increases continuously when it is more than 40. Thus, setting the number of category experts as 40 is reasonable for both the accuracy and the execution time.

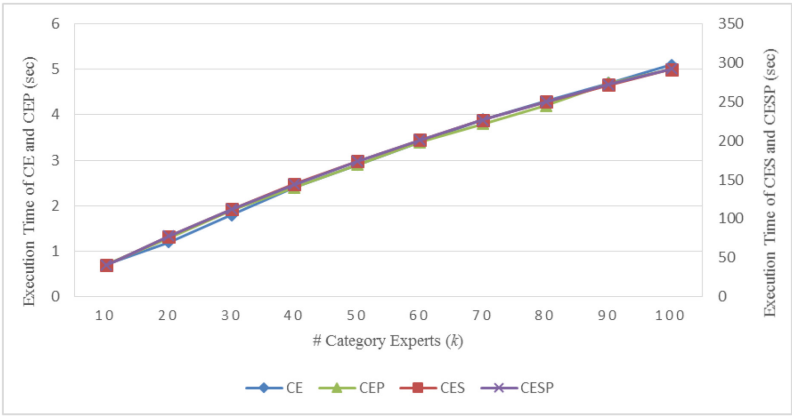
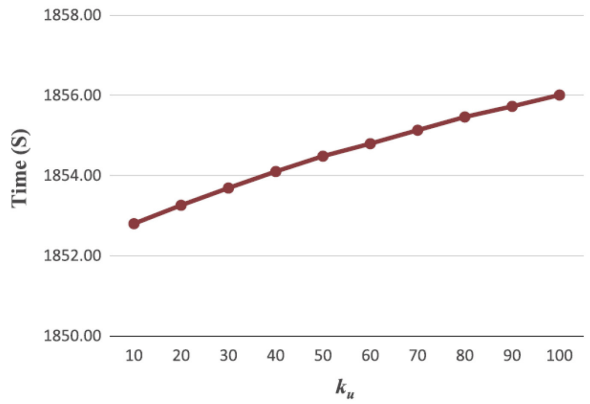
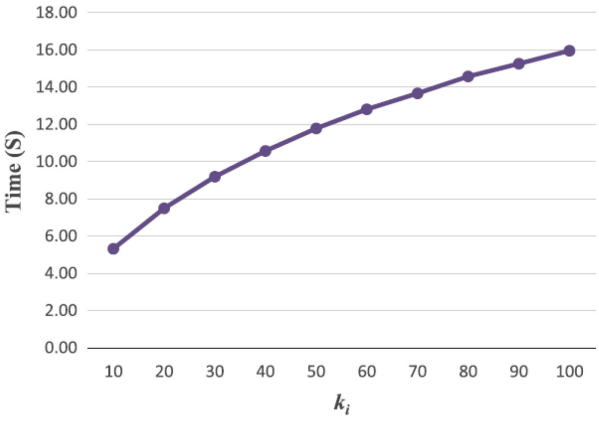


Figure 6.3. Execution time for CE, CES, CEP, and CESP.

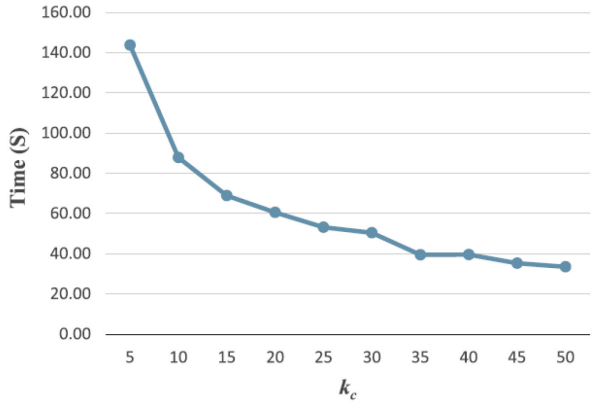
Figure 6.4 shows the execution time for UBM, IBM, and CBM using the Movielens dataset. In UBM, the more similar users there are, the longer the execution time is. The same amount of computing similarities is needed for all the active users regardless of the parameter k_u, so the difference between the execution times is not significant. IBM requires more time as the number of similar items increases. In CBM, the execution time decreases as the number of clusters increases. This is because the number of users in each cluster may decrease as the number of clusters increases.



a. User-based method



b. Item-based method



c. Clustering-based method

Figure 6.4. Execution time for UBM, IBM, and CBM.

In Table 6.5, we compare the average execution times of all the methods for predicting only one rating when it has the highest accuracy. This set of experiments is meaningful because each method has a different set of ratings that cannot be predicted. The proposed methods need much less

execution time than UBM. CE and CEP perform 32 times faster than UBM, and CES; CESP is 9 times faster. UBM requires the longest time because it calculates the similarities between the active user and all other users. The execution time for IBM is about 2.5 times longer than CE and CEP. This is because each method aggregates a different number of ratings for prediction (i.e., 80 ratings for CE, 60 ratings for CEP, and 100 ratings for IBM). IBM spends less execution time than CES and CESP because it does not calculate similarities in the rating prediction step. CBM requires more execution time than CE and CEP while it needs less execution time than CES and CESP. The reason is that the category experts tend to evaluate more items than other users, so more rating-comparisons are needed for computing a similarity value. All the methods need less execution time with the Ciao dataset because the dataset is relatively sparser than the Movielens dataset. We checked that the data sparsity considerably affects the time for similarity computations; the computation on Movielens (17.5us) needs more time than Ciao (1.7us).

Table 6.5. Execution time with parameter settings providing the best accuracy

Methods	**Execution time (*μs*)**	
	Movielens	**Ciao**
CE	92.2	7.6
CES	2852.8	252.8
CEP	93.6	8.2
CESP	3026.3	254.7
UBM	27561.0	5486.5
IBM	240.2	42.5
CBM	2117.7	437.0

Table 6.6 shows the coverage of the proposed methods with parameter *k* of their best accuracy using both Movielens and Ciao. The coverage represents how much percentage of unseen items is predictable with each method; the higher, the better [Her98, Mid04]. On Movielens, CE shows the highest coverage (98.19%) among all the proposed methods while CESP shows the lowest (94.31%). The coverages of CES and CEP are 97.93% and 94.55%, respectively. The coverage of CEP and CESP is lower than that of CE and CES since CEP and CESP cannot compute the category preference for a specific category when the active user has not evaluated any items in that category.

CESP outperforms IBM and CBM (up to 17% better than CBM) while it shows lower coverage

than UBM (4% worse than UBM) on Movielens. UBM shows higher coverage than IBM because the similarities of users would be more accurate than those of items. On Movielens, each user has more ratings (i.e., about 144.1) than each item has (i.e., about 66.2). CBM has the lowest coverage because most users who evaluated more items are classified into one cluster. We found that the difference between the average numbers of items evaluated by users in each cluster is up to about 40. Every result, excluding 5% ratings, shows lower coverage than those of using all ratings. The coverage using Ciao is worse than that using Movielens because Ciao is sparser than Movielens. In Ciao, CESP shows coverage 2 to 7 times better than the other methods.

Table 6.6. Coverage with parameter settings providing the best accuracy

Recommendation Methods	**Coverage**	
	Movielens	**Ciao**
CESP	94.31%	**27.36%**
UBM	**97.86%**	11.29%
IBM	87.81%	7.17%
IBM (excluding 5% ratings)	87.28%	5.05%
CBM	80.27%	4.27%
CBM (excluding 5% ratings)	80.11%	2.44%

In this section, we compared our methods with the existing methods in terms of accuracy, running time, and coverage. We observed that CESP, which combines all above approaches, leverages the tradeoff between the accuracy and execution time, and outperforms the existing methods in terms of coverage.

7. Conclusions

This chapter discussed how to improve the collaborative filtering in terms of accuracy, running time, and coverage. To achieve the goal, we proposed four approaches: (1) using the uninteresting items for any existing methods, (2) considering the trustors as well as trustees for trust-based methods, (3) imputing probable values for missing ratings using a trust network, and (4) using the category experts.

In the first approach, we observe that some unrated items could be also used to predict users' ratings if they are successfully recognized as uninteresting items. Based on this observation, we proposed a novel approach to unearth such uninteresting items by using a new notion of pre-use preferences in the borrowed OCCF framework and assign zero ratings to those items. This approach not only significantly augments a rating matrix with many zeros, which alleviates the data sparsity problem, but also prevents those uninteresting items from being recommended. Our approach is method-agnostic and thus can be easily applied to a wide variety of known CF methods. Through comprehensive experiments, we successfully demonstrated that our proposed approach is effective and practical, dramatically improving the accuracies of existing CF methods (e.g., item-based CF, SVD-based CF, and SVD++) by 2.5 to 5 times. Furthermore, our approach reduces the running time of those CF methods by 1.2 to 2.3 times when its setting produces the best accuracy.

In the second approach, we argue that the trustable users in $BWD(u)$ can be equally useful because a trustor and a trustee generally have similar interests. In this regard, $UND(u)$ in our proposal can be the largest set of trustable users who share interests with the active user u, and utilizing this set for the ratings prediction will give us chance to increase the coverage while maintaining the accuracy. In this chapter, we validated this argument through a series of experiments. We applied our approach to the existing methods and obtained positive results on our hypotheses. Especially, our approach has shown significant improvement in the coverage of ratings prediction for cold-start users.

The third approach imputes the unrated ratings through trust network firstly, and predicts the ratings via the matrix factorization technique. The imputation method imputes the unrated ratings by aggregating the ratings of the reliable neighbors who are found through the trust network. The reliable neighbors of a user are those users who are reachable from her through the bi-directional edges within the distance threshold in the trust network, and the approach of defining reliable neighbors is different from those existing methods. Moreover, the imputation method only considers

those unrated ratings of item which rated by enough number of reliable neighbors because the opinions of few users are difficult seen as reliable. Also, we performed the experiments based on three real-world data sets, and the result showed the proposed approaches are helpful to improve the accuracy. In those experiments, the proposed recommendation method overcomes the existing recommendation methods for the cold start users as well as the whole users.

In the fourth approach, we introduced a CE method that uses the notion of category experts in order to leverage the tradeoff between running time and accuracy in previous NBMs. We suggested the CES and CEP methods to improve the accuracy of the CE method, and finally combined all approaches to create the extended CESP method. By using the MovieLens and Ciao datasets, we confirmed that the 'similarity' between users and category experts and 'category interest' help increase the accuracy of the ratings prediction. In addition, the proposed CESP method showed accuracy close to the NBMs without any pre-computations of the similarity and model, and illustrated higher accuracy than CBM. By measuring the execution time for 100,000 sample items, we revealed that the CESP method performs 9 times faster than UBM and showed the maximum coverage among all existing methods.

Although this dissertation proposes novel approaches with higher accuracy and coverage as well as short running time, our approaches cannot deal with the other criteria such as serendipity and diversity. The serendipity indicates whether CF approaches can recommend those items that an active user has not expected, and the diversity indicates whether CF approaches can provide various kinds of items. In addition, our approaches cannot effectively remedy well-known problems, i.e., new user and item problems. The new user problem indicates that CF approaches cannot recommend items to new users who have not evaluated any items because it is impossible to analyze their preferences. Similar to that problem, the new item problem means that it is hard to recommend not-yet-rated items to anyone.

In the future, we are going to improve the proposed approaches to overcome the limitations mentioned above. One of possible solutions is to exploit other notions such as click, browsing, and purchase. We should analyze the relations between ratings and other notions and apply those notions to our approaches. The other solution is to apply content-based approach that effectively remedies the new item problem to our approaches; thus, we design hybrid approaches. Furthermore, to maximize the effect of our approaches, it is possible to integrate our approaches into one model. The integrated model would need short running time and have high accuracy and coverage by concurrently considering uninteresting items, trust networks, and category experts.

References

[Ado05] G. Adomavicius and A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," In *IEEE Transactions on Knowledge and Data Engineering*, *TKDE*, Vol. 17, No. 6, pp. 734–749, 2005.

[Ama09] X. Amatriain, et al. "The Wisdom of the Few: A Collaborative Filtering Approach based on Expert Opinions from the Web," In *Proc. of the 32nd Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval*, *SIGIR*, pp. 532–539, 2009.

[Ank99] M. Ankerst, et al., "OPTICS: Ordering Points to Identify the Clustering Structure," In *ACM Sigmod Record,* Vol. 28, No. 2, pp. 49-60, 1999.

[Bel07] R. Bell and Y. Koren, "Lessons from the Netflix Prize Challenge," In *ACM SIGKDD Explorations Newsletter*, Vol. 9, No. 2, pp. 75–79, 2007.

[Bil98] D. Billsus and M. Pazzani, "Learning Collaborative Information Filters," In *Proc. 15th Int'l Conf. on Machine Learning*, *ICML*, pp. 46-54, 1998.

[Bri98] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," In Computer Networks and ISDN Systems, Vol. 30, No. 1-7, pp. 107-117, 1998.

[Bre98] J. Breese et al. "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," In *Proc. of the 14th Conf. on Uncertainty in Artificial Intelligence, UAI*, pp. 43–52, 1998.

[Cho07] J. Cho, K. Kwon, and Y. Park, "Collaborative Filtering Using Dual Information Sources," In *IEEE Intelligent Systems,* Vol. 22, No. 3, pp. 30–38, 2007.

[Cre10] P. Cremonesi et al., "Performance of Recommender Algorithms on Top-N Recommendation Tasks," In *Proc. of the 4th ACM Conf. on Recommender Systems, RecSys*, pp. 39–46, 2010.

[Dal13] N. Dalvi et al., "Para'normal' Activity: On the Distribution of Average Ratings," In *Proc. of the 21st Int'l AAAI Conf. on Weblogs and Social Media*, *ICWSM,* pp. 110–119, 2013.

[Del99] J. Delgado and N. Ishii, "Memory-based Weighted-Majority Prediction for Recommender Systems," In *Proc. of ACM SIGIR Workshop Recommender Systems: Algorithms and Evaluation*, 1999.

[Est96] M. Ester et al., "A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," In *Proc. of Int'l Conf. on Knowledge Discovery and Data Mining*, *KDD*, pp. 226–231, 1996.

[Gan11] Z. Gantner et al., "Mymedialite: A Free Recommender System Library," In *Proc. of the 5th*

ACM Conf. on Recommender Systems, RecSys, pp. 305–308, 2011.

[Gol05] J. Golbeck, *Computing and Applying Trust in Web-based Social Networks*, Ph. D. Dissertation, University of Maryland, College Park, USA, 2005.

[Ha12] J. Ha et al. "Top-N Recommendation through Belief Propagation," In *Proc. of the 21st ACM Int'l Conf. on Information and Knowledge Management, CIKM*, pp. 2343–2346, 2012.

[Han01] J. Han and M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufman, 2001.

[Han14] G. Han et al., "Management and Applications of Trust in Wireless Sensor Networks: A Survey," In *Journal of Computer and System Sciences*, Vol. 80, No. 3, pp. 602–617, 2014.

[Her99] J. L. Herlocker et al., "An Algorithmic Framework for Performing Collaborative Filtering," In *Proc. of the 22nd Int'l Conference on Research and Development in Information Retrieval, SIGIR*, pp. 230–237, 1999.

[Her04] J. L. Herlocker et al., "Evaluating Collaborative Filtering Recommender Systems," In *ACM Transactions on Information Systems*, *TOIS*, Vol. 22, No. 1, pp. 5-53, 2004.

[Hof04] T. Hofmann, «Latent Semantic Models for Collaborative Filtering," In *ACM Transactions on Information Systems*, *TOIS*, Vol. 22, No. 1, pp. 89-115, 2004.

[Hu09] N. Hu et al. "Overcoming the J-Shaped Distribution of Product Reviews," In *Communications of the ACM*, Vol. 52, No. 10, pp. 144–147, 2009.

[Hua04] Z. Huang, H. Chen, and D. Zeng, "Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering," In *ACM Transactions on Information Systems*, *TOIS*, Vol. 22, No. 1, pp. 116–142, 2004.

[Jam09] M. Jamali and M. Ester, "TrustWalker: A Random Walk Model for Combining Trust-based and Item-based Recommendation," In *Proc. ACM Int'l. Conf. on Knowledge Discovery and Data Mining, KDD,* pp. 397-406, 2009.

[Jam10] M. Jamali and M. Ester, "A Matrix Factorization Technique with Trust Propagation for Recommendation in Social Networks," In *Proc. of the 4th ACM Conf. on Recommender Systems*, *RecSys*, pp.135–142, 2010.

[Kar01] G. Karypis, "Evaluation of Item-based Top-N Recommendation Algorithms," In *Proc. of the 10th Int'l Conf. on Information and Knowledge Management*, *CIKM*, pp. 247–254*,* 2001.

[Kor08] Y. Koren, "Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model," In *Proc. ACM Int'l. Conf. on Knowledge Discovery and Data Mining, KDD,* pages 426–434, 2008.

[Kor09] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender

Systems," In *IEEE Computer*, Vol. 42, No. 8, pp. 30–37, 2009.

[Lin03] G. Linden, B. Smith, and J. York, "Amazon.com Recommendations: Item-to-Item Collaborative Filtering," In *IEEE Internet Computing,* Vol.7, No. 1, pp. 76-80, 2003.

[Liu10] J. Liu et al., "Personalized News Recommendation based on Click Behavior," In *Proc. of the 15th Int'l Conf. on Intelligent User Interfaces, IUI*, pp. 31–40, 2010.

[Liu11] Q. Liu, B. Cheng, and C. Xu, "*Collaborative Filtering based on Star Users*," In *Proc. of the 23rd IEEE Int'l Conf. on Tools with Artificial Intelligence*, *ICTAI*, pp. 223–228*,* 2011.

[Ma07] H. Ma, I. King, and M. R. Lyu, "Effective Missing Data Prediction for Collaborative Filtering," In *Proc. of the 30th Annual Int'l Conf. on Research and Development in Information Retrieval, SIGIR*, pp. 39–46, 2007.

[Ma08] H. Ma et al., "Sorec: Social Recommendation Using Probabilistic Matrix Factorization," In *Proc. of the 17th ACM Int'l Conf. on Information and Knowledge Management, CIKM*, pp. 931-940, 2008.

[Ma09a] H. Ma, M. R. Lyu, and I. King, "Learning to Recommend with Trust and Distrust Relationships," In *Proc. of the 3rd ACM Int'l Conf. on Recommender Systems, RecSys*, pp. 189-196*,* 2009.

[Ma09b] H. Ma, I. King, and M. R. Lyu, "Learning to Recommend with Social Trust Ensemble," In *Proc. of the 32nd Int'l Conf. on Research and Development in Information Retrieval, SIGIR*, pp. 203–210, 2009.

[Ma10] H. Ma, I. King, and M. R. Lyu, "Learning to Recommend with Social Trust Ensemble," In *Proc. of the 32nd Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval*, *SIGIR*, pp. 203–210*,* 2010.

[Mar09] B. Marlin and R. Zemel, "Collaborative Prediction and Ranking with Non-Random Missing Data," In *Proc. of the 3rd ACM Conf. on Recommender Systems*, *RecSys*, pp. 5–12, 2009.

[Mas04] P. Massa and B. Bhattacharjee, "Using Trust in Recommender Systems: An Experimental Analysis," In *Proc. of 2nd Int'l Conf. on Trust Management*, pp. 221-235, 2004.

[Mas05] P. Massa and P. Avesani, "Controversial Users Demand Local Trust Metrics: An Experimental Study on Epinions.com Community," In *Proc. of the National Conf. on Artificial Intelligence*, Vol. 20, No. 1, pp. 121-126, 2005.

[Mas07] P. Massa and P. Avesani, "Trust-aware Recommender Systems," In *Proc. of the 1st ACM Conf. on Recommender Systems*, *RecSys*, pp. 17-24, 2007.

[Mid04] S. Middleton, N. Shadbolt, and D. D. Roure, "Ontological User Profiling in Recommender Systems," In *ACM Transactions on Information Systems, TOIS,* Vol. 22, No. 1, pp. 54–88,

2004.

[Mil67] Stanley Milgram, "The Small World Problem," *Psychology Today*, Vol. 2, No. 1, pp. 61-67, 1967.

[Mil03] B. N. Miller et al., “Movielens Unplugged: Experiences with an Occasionally Connected Recommender System,” In *Proc. of Int’l Conf. on Intelligent User Interfaces*, *IUI*, pp. 263–266*,* 2003.

[Miy00] K. Miyahara and M. Pazzani, “Collaborative Filtering with the Simple Bayesian Classifier,” In *Proc. of the 6th Pacific Rim Int’l Conf. on Artificial Intelligence*, pp. 679–689, 2000.

[Moh09] M. Jamali and M. Ester, “A Random Walk Model for Combining Trust-based and Item-based Recommendation,” In *Proc. of the ACM Int’l Con. On Knowledge Discovery and Data Mining, KDD,* pp. 397-406, 2009.

[Mog11] S. Moghaddam, M. Jamali, and M. Ester, "Review Recommendation: Personalized Prediction of the Quality of Online Reviews," In *Proc. of the 20th ACM Int’l Conf. on Information and Knowledge Management, CIKM*, pp. 2249-2252*,* 2011.

[Nak98] A. Nakamura and N. Abe, “Collaborative Filtering Using Weighted Majority Prediction Algorithms,” In *Proc. of 15th Int’l Conf. on Machine Learning, ICML*, pp. 395–403, 1998.

[Ngu14] T. T. Nguyen et al., “Exploring the Filter Bubble: the Effect of Using Recommender Systems on Content Diversity,” In *Proc. of the 23rd Int’l Conf. on World Wide Web, WWW*, pp. 677–686, 2014.

[Niw06] S. Niwa et al., “Web Page Recommender System based on Folksonomy Mining for ITNG ’06 Submissions,” In *Proc. of the 3rd Int’l Conf. on IEEE Information Technology: New Generations, ITNG*, pp. 383–393, 2006.

[Pan08] R. Pan et al., “One-Class Collaborative Filtering,” In *Proc. of the 8th Int’l Conf. on Data Mining, ICDM*, pp. 502–511, 2008.

[Pha13] X. H. Pham, J. J. Jung, and N. T. Nguyen, “Integrating Multiple Experts for Correction Process in Interactive Recommendation Systems,” In *Computational Collective Intelligence. Technologies and Applications*, Vol. 19, No. 4, pp. 581–599, 2013.

[Pha14] X. H. Pham, et al., “<a, v>-Spear: A New Method on Expert-based Recommendation Systems,” In *Cybernetics and Systems*, Vol. 45, No. 1, pp. 165– 179, 2014.

[Pop01] A. Popescul et al., “Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments,” In *Proc. of the Conf. on Uncertainty in Artificial Intelligence, UAI*, pp. 437-444, 2001.

[Ren94] P. Resnick et al., “Grouplens: An Open Architecture for Collaborative Filtering of Netnews,”

In *Proc. of the ACM Conf. on Computer Supported Cooperative Work*, pp. 175–186, 1994.

[Ren05] J. Rennie and N. Srebro. "Fast Maximum Margin Matrix Factorization for Collaborative Prediction," In *Proc. of the 22nd Int'l Conf. on Machine Learning, ICML*, pp. 713–719, 2005.

[Ren12] Y. Ren et al., "The Efficient Imputation Method for Neighborhood-based Collaborative Filtering," In *Proc. of the 21st ACM Int'l Conf. on Information and Knowledge Management, CIKM*, pp. 684–693, 2012.

[Res94] P. Resnick et al., "Grouplens: An Open Architecture for Collaborative Filtering of Netnews," In *Proc. of the ACM Conf. on Computer Supported Cooperative Work*, pp. 175–186, 1994.

[Sal07a] R. Salakhutdinov et al., "Restricted Boltzmann Machines for Collaborative Filtering," In *Proc. of the 24th Int'l Conf. on Machine Learning, ICML*, pp. 791–798, 2007.

[Sal07b] R. Salakhutdinov and A. Mnih, "Probabilistic Matrix Factorization," In *Advances in Neural Information Processing Systems, NIPS*, pp. 1257–1264, 2007.

[Sal08] R. Salakhutdinov and A. Mnih, "Bayesian Probabilistic Matrix Factorization Using Markov Chain Monte Carlo," In *Proc. of the 25th Int'l Conf. on Machine learning, ICML*, pp. 880–887, 2008.

[Sar00] B. Sarwar et al., Application of Dimensionality Reduction in Recommender Systems—A Case Study, July, 2000. Minnesota Univ. Minneapolis Dept. Of Computer Science (No. TR-00-043).

[Sar01] B. Sarwar et al., "Item-based Collaboration Filtering Recommendation Algorithms," In *Proc. of the 19th Int'l Conf. on World Wide Web*, *WWW,* pp. 285–295, 2001.

[Sar02] B. M. Sarwar et al., "Recommender Systems for Large-Scale E-commerce: Scalable Neighborhood Formation Using Clustering," In *Proc. of the 5th Int'l Conf. on Computer and Information Technology*, *ICCIT*, 2002.

[Sch02] A. Schein et al., "Methods and Metrics for Cold-Start Recommendations," In *Proc. of the Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval, SIGIR,* pp. 253-260, 2002.

[Sha11] B. Shapira, Recommender Systems Handbook, Springer, 2011.

[Sin09] V. Sindhwani et al., "A Family of Non-Negative Matrix Factorization for One-Class Collaborative Filtering," In *Proc. of the 3rd ACM Conf. on Recommender Systems*, *RecSys*, 2009.

[Sob99] I. Soboroff and C. Nicholas, "Combining Content and Collaboration in Text Filtering," In *Proc. of the Int'l Joint Conf. on Artificial Intelligence Workshop: Machine Learning for*

Information Filtering, pp. 86–91, 1999.

[Som01] G. Somlo and A. Howe, "Adaptive Lightweight Text Filtering," In *Proc. of the Int'l Symp. Inteeligent Data Analysis, IDA,* pp. 319-329, 2001.

[Sre03] N. Srebro and T. Jaakkola, "Weighted Low-Rank Approximations," In *Proc. of the 20th Int'l Conf. on Machine Learning, ICML*, pp. 720–727, 2003.

[Ste10] H. Steck, "Training and Testing of Recommender Systems on Data Missing Not at Random," In *Proc. of Int'l Conf. on Knowledge Discovery and Data Mining*, *KDD*, pp. 713–722, 2010.

[Su09] X. Su and T. Khoshgoftarr, "A Survey of Collaborative Filtering Techniques," In *Advances in Artificial Intelligence,* 2009.

[Tan12] J. Tang et al.,."eTrust: Understanding Trust Evolution in an Online World," In *Proc. of the 18th Int'l Conf. on Knowledge Discovery and Data Mining, KDD*, pp. 253–261, 2012.

[Ung98] L. H. Ungar and D. P. Foster, "Clustering Methods for Collaborative Filtering," In *Proc. of AAAI Workshop on Recommendation Systems*, pp. 114–129, 1998.

[Yin15] H. Yin et al. "Modeling Location-based User Rating Profiles for Personalized Recommendation," In *ACM Transactions on Knowledge Discovery from Data, TKDD*, Vol. 9, No. 3, pp. 19, 2015.

[Yu11] L. Yu, R. Pan, and Z. Li, "Adaptive Social Similarities for Recommender Systems," In *Proc. of the 5th ACM Int'l Conf. on Recommender Systems, RecSys*, pp. 257-260, 2011.

[Yun01] L. Yun et al., "Improving Rating Estimation in Recommender Using Demographic Data and Expert Opinions," In *Proc. of IEEE 2nd Int'l Conf. on Software Engineering and Service Science*, pp. 120–123, 2001.

[Zha01] Y. Zhang and J. Callen, "Maximum Likelihood Estimation for Filtering Thresholds," In *Proc. of the Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval, SIGIR,* pp. 294-302, 2001.

[Zha02] Y. Zhang, J. Callan, and T. Minka, "Novelty and Redundancy Detection in Adaptive Filtering," In *Proc. of the Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval, SIGIR,* pp. 81-88, 2002.

[Zha05] S. Zhang et al., "Using Singular Value Decomposition Approximation for Collaborative Filtering," In *Proc. of 7th IEEE Int'l Conf. on E-Commerce Technology*, pp. 257–264, 2005.

Abstract

Exploiting Uninteresting Items, Trust Networks, and Category Experts for Effective Recommendation Systems

Won-Seok Hwang
Dept. of Electronics and Computer Engineering
The Graduate School
Hanyang University

The recommendation system is a technique that provides an active user with a few items that she would like. Among various recommendation systems, the collaborative filtering (CF) is one of the most popular and effective techniques. The CF approaches recommend items given with high ratings by neighbors who have taste similar to the active user based on users' ratings. Meanwhile, most users evaluate only a few items, leading to the data sparsity problem, which occurs low accuracy and coverage in the CF approaches. In addition, as the numbers of users, items, and ratings increase, the time for analysis and prediction gets longer in CF approaches.

To improve accuracy and performance of the CF approaches, this dissertation proposes four approaches with three notions: (1) uninteresting items, (2) trust networks, and (3) category experts. The uninteresting items indicate those items that are not attractive and thus unlikely to be purchased or used by users. Because the users do not give ratings to those uninteresting items, the existing CF approaches ignore them in their recommendations. Our first approach improves the accuracy by applying the notion of uninteresting items to the framework of any existing CF approaches. In addition, it improves the performance by reducing the number of items whose preferences have to be predicted because it completely prevents those uninteresting items from being recommended as top-N items.

The trust network is a kind of social network implying the trust relationships between users. Several existing approaches examine the trust network to find users who have taste similar to an active user. They assume that only trustees (i.e., users whom a certain user trusts) are similar to the active user, but we think trustors (i.e., users who trust a certain user) also are similar to her. Based on this assumption, our second approach uses trustors as well as trustees, and improves the accuracy and coverage in the CF approaches.

Furthermore, by extending this idea, we propose a novel imputation approach that exploits the trust network to enrich the collaborative information. Unlike the existing approaches that only utilize the users' ratings, our approach additionally utilizes the trust network implying the information different from the ratings. In addition, it imputes only some missing ratings whose values are likely to be inferred correctly rather than all missing ratings. Thus, because of careful imputation with abundant information, our approach improves accuracy of the CF approaches.

One of the bottleneck of CF approach is finding neighbors of an active user. To alleviate this performance problem, our fourth approach utilizes the concept of category experts who have a large amount of knowledge in their own category. Our approach defines the category experts as those users who evaluate items more than others. Thus, using the category experts, instead of neighbors, is more efficient in terms of performance because finding (or maintaining category) experts needs a computational cost much less than finding (or maintaining) neighbors. Furthermore, our approach produces accurate results because the users would follow the opinions of experts in a real world.

Through comprehensive experiments with several real-world dataset, this dissertation demonstrates that our approaches exploiting uninteresting items and trust networks improve the accuracies and coverages compared to existing CF approaches. The fourth approach is more efficient in terms of execution time because maintaining category experts needs a computational cost much less than finding (or maintaining) neighbors.

감사의 글

박사 과정 동안 연구를 진행하면서, 많은 분들에게 크나큰 도움을 받았고, 이 분들의 도움으로 박사 학위를 받을 수 있었습니다. 감사한 마음을 모두 표현하기 어렵겠지만, 짧은 감사의 글로 그 마음을 조금이나마 표현하려 합니다.

먼저, 많은 관심과 격려로 지도해주신 김상욱 교수님께 진심으로 감사 드립니다. 업무나 삶의 자세에서 부족하고 어리석은 점이 많았던 저를 포기하지 않고 이끌어 주셨기에 지금의 제가 있는 것 같습니다. 교수님께서 말씀하신 조언과 보여주신 열정적인 삶의 모습을 가슴에 새겨 앞으로도 더욱 발전할 수 있도록 노력 하겠습니다.

학위 논문의 심사를 위해 바쁘신 와중에도 귀중한 시간을 내어주신 차재혁 교수님, 박희진 교수님, 강수용 교수님, 그리고 펜실베니아 주립 대학의 이동원 교수님께도 감사 드립니다. 또한, 박사 과정 기간 동안 연구에 많은 조언을 주신 이기천 교수님, 원영준 교수님, 그리고 카이스트의 최호진 교수님, 한국 외국어 대학교의 이종욱 교수님께도 감사 드립니다. 교수님들의 조언이 있었기에 더 나은 연구자로써 성장할 수 있었습니다.

대학원 생활 내내 대부분의 시간을 함께 보낸 데이터 및 지식 공학 연구실 선배님과 후배들에게도 고맙다는 말을 전하고 싶습니다. 박사 과정으로써 모범을 보여주시고 많은 도움을 주셨던 승환 선배, 민희 선배, 석호 선배, 상철 선배, 덕호 선배, 영주 선배, 종대 선배, 용석 선배, 지행 선배에게 감사의 마음 전합니다. 또한, 동기로 석사 과정 동안 어려움을 함께 했던 두열 형, 정환 형, 세미 누나에게도 고마움을 전합니다. 그리고 함께 졸업하는 지운, 현교, 창욱, 교환, 태희와 앞으로 연구실을 더욱 빛내줄 Masoud, 지원, 용연, 동규, 윤석, 윤용, Irfan, Suriana, 연창, 형욱, 준호, 재근, 영남, 희정, 유진, 나영에게도 앞으로 좋은 일들이 가득하길 빌겠습니다. 특히, 부족한 사형임에도 믿고 따라주었던 수민, 호종 형, 소우, 정, 창욱, 주안, 영남에게도 고맙다는 말을 전하고 싶습니다.

마지막으로 언제나 저를 믿고 응원해 주신 부모님께 감사 드립니다. 그리고 멀리서 응원해 준 동생에게도 고맙다는 말 전하고 싶습니다. 그리고 힘들고 좌절할 때마다

큰 힘이 되어준 성이에게도 고마움을 전합니다. 사랑하는 가족과 연인이 있었기에 힘든 박사 과정을 이겨내고, 학위라는 결실을 얻을 수 있었습니다.

이 밖에도 자세히 언급하지 못하였지만, 물심양면으로 도움을 준 고등학교 및 대학 친구들과 저를 아시는 모든 분들께 감사의 마음을 전합니다.

2016년 2월

황 원 석

연구 윤리 서약서

본인은 한양대학교 대학원생으로서 이 학위논문 작성 과정에서 다음과 같이 연구 윤리의 기본 원칙을 준수하였음을 서약합니다.

첫째, 지도교수의 지도를 받아 정직하고 엄정한 연구를 수행하여 학위논문을 작성한다.

둘째, 논문 작성시 위조, 변조, 표절 등 학문적 진실성을 훼손하는 어떤 연구 부정행위도 하지 않는다.

셋째, 논문 작성시 논문유사도 검증시스템 "카피킬러"등을 거쳐야 한다.

2015년12월23일

학위명 : 박사

학과 : 전자컴퓨터통신공학과

지도교수 : 김상욱

성명 : 황원석 (서명)

한 양 대 학 교 대 학 원 장 귀 하

Declaration of Ethical Conduct in Research

I, as a graduate student of Hanyang University, hereby declare that I have abided by the following Code of Research Ethics while writing this dissertation thesis, during my degree program.

"First, I have strived to be honest in my conduct, to produce valid and reliable research conforming with the guidance of my thesis supervisor, and I affirm that my thesis contains honest, fair and reasonable conclusions based on my own careful research under the guidance of my thesis supervisor.

Second, I have not committed any acts that may discredit or damage the credibility of my research. These include, but are not limited to : falsification, distortion of research findings or plagiarism.

Third, I need to go through with Coppykiller Program(Internet-based Plagiarism-prevention service) before submitting a thesis."

DECEMBER 23, 2015

Degree :	Doctor
Department :	DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
Thesis Supervisor :	Sang-Wook Kim
Name :	HWANG WON-SEOK (Signature)