Improving Existing Collaborative Filtering Recommendations via Serendipity-Based Algorithm

Yongjian Yang, Yuanbo Xu, En Wang, Jiayu Han, and Zhiwen Yu, *Senior Member, IEEE*

***Abstract*—In this paper, we study how to address the sparsity, accuracy and serendipity issues of top-N recommendation with collaborative filtering (CF). Existing studies commonly use rated items (which form only a small section in a rating matrix) or import some additional information (e.g., details about the items and users) to improve the performance of CF. Unlike these methods, we propose a novel notion towards a huge amount of unrated items: serendipity item. By utilizing serendipity items, we propose concise satisfaction and interest injection (CSII), a method that can effectively find interesting, satisfying, and serendipitous items in unrated items. By preventing uninteresting and unsatisfying items to be recommended as top-N items, this concise-but-novel method improves accuracy and recommendation quality (especially serendipity) substantially. Meanwhile, it can address the sparsity and cold-start issues by enriching the rating matrix in CF without additional information. As our method tackles rating matrix before recommendation procedure, it can be applied to most existing CF methods, such as item-based CF, user-based CF and matrix factorization-based CF. Through comprehensive experiments using abundant real-world datasets with LensKit implementation, we successfully demonstrate that our solution improves the performance of existing CF methods consistently and universally. Moreover, comparing with baseline methods, CSII can extract uninteresting items more carefully and cautiously, avoiding potential items inferred by mistake.**

***Index Terms*—Collaborative filtering, unrated items, serendipitous recommendation, matrix factorization.**

I. Introduction

RECOMMENDER Systems (RSs) can be commonly defined as programs which attempt to recommend the most valuable items (products or services) to special kinds of users (individuals or businesses) by extracting a user's preference from various related information of the users, the items and the relationships between them [1]. The target of developing RSs is to tackle most relevant information from a huge amount of data, thereby providing customized services, finding substantial recommendations [2]. The most important feature of a recommender system is its ability to infer a user's preferences of interests and satisfactions by analyzing behaviors of this user and/or behaviors of other users to generate customized recommendations [3]. RSs usually make use of different aspects of information to provide users with the predictions and recommendations of items [4]. They try to balance the factors like accuracy, novelty, dispersity and stability in recommendation results and to reduce storage, computation complexity in recommend procedures. Collaborative Filtering (CF) methods play a significant role in RSs, though they are often used along with other filtering techniques like content-based, knowledge-based or social ones [2], [4]–[6].

TABLE I
A User-Item Rating Matrix

	Item 1	Item 2	Item 3	Item 4
User 1	1	3	5	3
User 2	?	2	?	?
User 3	5	?	?	?
User 4	?	2	4	?

The basic idea of Collaborative Filtering is the process of filtering or evaluating items for a user with the influence of other users' ratings [7], and recommend proper items to users (shown in Table I). There are two major challenges in Collaborative Filtering [8]. Firstly, how to tackle the sparsity of ratings in User-Item matrix? As mentioned before, ratings r_{ui} (user u's rating on item i) in User-Item matrix are the direct feedbacks of users' opinion. However, due to the popularity and exponential growth of e-commerce websites (e.g., Amazon(*www.amazon.com*), Alibaba(*www.taobao.com*)) and online streaming websites (e.g., Netflix(*www.netflix.com*), Youku(*www.youku.com*)), the magnitudes of items in User-Item Matrix are growing faster than the users. Existing CF methods attempt to exploit the ratings that users have provided, but may become less effective when the proportion of known ratings in a user-item matrix is quite small. For a user-item matrix R with m users and n items, usually $n \gg m$ [9]. The sparsity of ratings results in the Cold-Start and Lower accuracy problems in CF [10], [11]. As such, we believe the sparsity is a core problem to improve the performance of CF. Secondly, it is quite hard to infer the preference of users only by ratings. The ratings are sparse in user-item matrix. And users' interests are constantly changing along with time, social effects

Manuscript received May 31, 2017; revised August 22, 2017 and October 5, 2017; accepted November 20, 2017. Date of publication December 1, 2017; date of current version June 15, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61272412, in part by the Jilin Scientific and Technological Development Program under Grant 20160204021GX, and in part by the Graduate Innovation Fund of Jilin University under Grant 2016184. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Elisa Ricci. *(Corresponding author: En Wang.)*

Y. Yang, Y. Xu, E. Wang, and J. Han are with the Department of Computer Science and Technology, Jilin University, Changchun 130012, China (e-mail: yyj@jlu.edu.cn; yuanbox15@mails.jlu.edu.cn; wangen@jlu.edu.cn; jyhan15@mails.jlu.edu.cn).

Z. Yu is with the School of Computer Science, Northwestern Polytechnical University, Xian 710072, China (e-mail: zhiwenyu@nwpu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2017.2779043

and other complex factors. Several existing works have used additional information, such as satisfactions for items [5], social relationships [12], location information [13] and cross-domain knowledge [14], to infer users' interests. But using these additional information cause some other problems. For example, they need additional computing power, storage and computing time of recommendation system.

Our method is inspired by [9], and also based on the hypothesis in CF.

Hypothesis: *Filling the empty cells in user-item matrix (i.e., unrated items) can improve the accuracy, quality and computing time of the top-N recommendation by CF methods.*

There are some ideas based on **Hypothesis** in [9], such as 1) exploit uncharted unrated items, 2) some unrated items are uninteresting and 3) uninteresting items have low pre-use preferences. Our work is quite different from [9] because: As mentioned before, users' interest can be affected by many factors, especially the satisfaction for items they have rated. [9] uses r_{ui} (user u's rating on item i) to infer the interests independently, and assumes that users' preferences always follow the items they have rated, no matter ratings are high or low. But they ignore the fact that satisfactions of rated items will indicate the interest of users. For example, Jack buys a cap and a jacket from Amazon, he loves the cap so much and ranks it 5 (highest level), but he dislikes the jacket so he ranks it 1 (lowest level). For methods in [9](and some implicit feedback methods like eALS), both the cap and the jacket will be marked as interesting items of I^t_{Jack} as Jack's preference. And recommendations of [9] for Jack may be still caps he likes and jackets he dislikes. Our work notices the important phenomenon in real world, and uses two novel concepts: $pre-interest$ and $post-satisfaction$ to infer users' preference better than [9] does.

Otherwise, $serendipity$ is a core factor to evaluate the recommender system [2], [15]. A serendipitous item for a user should meet certain requirements. A user is likely to consider an item irrelevant at first sight if the item is significantly different from his taste, which is unexpected [16]. However, the item is supposed to become interesting for the user eventually. In a word, a serendipity is an item that users' interests are low before, but satisfactions are high after using, buying or other contact actions with it. Our method over-performs [9] in finding users' serendipity items because we have filtered unrated items more critically and carefully.

Based on the ideas above, we propose **Concise Satisfaction and Interest Injection (CSII)**, a novel method to improve CF methods with respect to sparsity problem and preference inferring problems with only user-item matrix, rather than additional information. To the best of our knowledge, this is the first ever attempt to combine unsatisfying and uninteresting items in unrated items together to tackle the sparsity and preference problems without additional information. Our proposal is $methodological$ so that it can be applied to a wide variety of CF techniques seamlessly, improving their performance in running time and recommendation quality, especially serendipity, and finding potential recommendations.

The main contributions of this paper are as follows:

1) We propose a novel notion: $Serendipity\ Item$, to tackle the sparsity and accuracy problems in recommendation system. Firstly, We define $post-satisfaction$ and $pre-interest$ in user-item matrix. With these two categories, we define $Serendipity\ Item$, that users' interests are low before, but satisfactions are high after using, buying them.
2) We propose a novel method: Concise Satisfaction and Interest Injection (CSII) to extract users' preference in unrated items, and find potential recommendations. We formulate the extraction problem into one-class collaborative filtering (OCCF) problems, and use a novel Matrix Factorization (MF) based method to solve them. Our proposed method is called before recommendation process, so it can be applied in many popular recommender systems to improve the performance.
3) We conduct extensive experiments on real-world datasets, in which the encouraging results demonstrate that our proposed method a) eases the sparsity of dataset, b) obtains lower errors and running time applied in existing CF methods (item-based and MF-based) and c) improves the serendipity performance of recommender system, and finds potential recommendations.

The rest of the paper is organized as follows: In the next section, we give a brief introduction to the related works. In Section III, we present some basic definitions of our approach. We introduce the details of our approach in Section IV and analyze the optimal performance of CSII in Section V. In Section VI, we conduct experiments to evaluate our proposed methods and give an analysis on the experimental results. Finally, we conclude with a discussion on our work in Section VII.

II. Related Work

Generally, all CF methods can be categorized into two different types: $model-based$ CF methods and $memory-based$ CF methods [1]. $Memory-based$ CF focus on users' ratings on rated items to predict ratings for unrated items [3], [5]. The common way to predict ratings is using similarity between different items to build relationships over rated and unrated items, which is called item-based methods [17]. While $model-based$ CF usually employ matrix decomposition to tackle recommendation issues [18]. They treat ratings as a user-item matrix, which entries are ratings (for rated items) and NULL (for unrated items), and try to fill all the NULLs as the predicted ratings for unrated items with mathematical methods [19]. The most representative model-based method is SVD, SVD++ [7], [13], [16], [18].

The most important issues in CF are accuracy, data sparsity and cold-start [7], [8]. There are many improved methods based on both basic methods introduced above [9]–[13]. Meanwhile, it is very popular to use different additional information to improve accuracy and efficiency of CF recommendation [20], [21]. [10] connects social media, users' action and microblog information to tackle cold start issues. Sun [22], Han [23] utilize video information to make personalized recommendations. [11] employs users' context to improve accuracy of recommendation [6], [24], while some methods learn users' preference in their social informations [12], [13]. Several researchers focus on spatio-temporal recommendation [25]–[27]. Recently,

recommendation is combined with neural computing [28], [29]. These methods can improve the performance of recommendation in different aspects with different extents. However, the numbers of items and users in recommender system is quite large [1], [5], [20], [21], and it costs a lot storage and compute ability to utilize users' additional information to improve accuracy. [9] uses mathematical methods (OCCF [30]) to extract users' interest for unrated items without additional information to improve the accuracy and tackle data sparsity issues. But there is a core problem for the proposed method in [9]: it can only find the same kind of items users have rated, but not the items that maybe unknown satisfying with, which we called $serendipity$.

$Serendipity$ is an important indicator to evaluate the recommender system, which has attracted more and more attentions of recommender system researchers [2], [15], [16]. A serendipitous item for a user should meet certain requirements. A user is likely to consider an item irrelevant at first sight if the item is significantly different from his/her taste (unexpected) [16]. Meanwhile, the item is required to be relevant to eventually become interesting for the user [15]. If a recommender system ignores the serendipity, all users in this system will find that the recommendation is only what they have bought or scanned before, with no surprise [2]. In a word, a serendipity is an item in which users' interests are low before, but satisfactions are high after using, buying or other actions on it.

TABLE II
THE EFFECT OF PRE-INTEREST AND POST-SATISFACTION FOR r_{ui}

	post-satisfaction	pre-interest	ratings
User4 to Item1	unknown	low or unknown	?
User4 to Item2	low	high	2
User4 to Item3	high	high	4

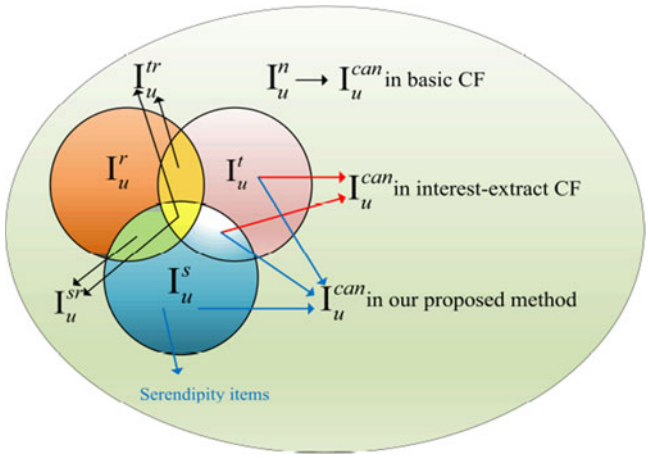


Fig. 1. Relationships between different categories of items in recommeder system.

III. PRELIMINARIES

In a recommender system, let U be a set of m users $U = \{u_1, u_2 ... u_m\}$, and I be a set of n items $I = \{i_1, i_2 ... i_n\}$. r_{ui} means the rating user u marked for item i. So we build a user-item matrix $R_{m \times n}$, whose entry are r_{ui} for rated items and $null$ for unrated items.

Before introducing our definitions, we should analyze rated items and unrated items in user-item matrix. The motivation for users to use or buy an item is that users are very interested in it when they saw it in web-pages, heard about it from their friends or any other ways. We define all the rated items of user u as u's interesting items (I_u^{tr}). However, users will not accept new items of same category of rated ones if they are not satisfied with what they used or bought. Luckily, we have all the ratings for all the rated items. So we can find out which ones are satisfactions according to user's preference. So we define the high ranks items as satisfying items (I_u^{sr}).

There are two situations in unrated items of the user u. First, users may not browse these items in web pages, or receive any information about them from other ways. In the real world, this is the most common situation. Second, u will not use or buy the items which he shows no interest. However, in this situation, u will lose the chance to accept serendipity item. In this paper, we can infer the satisfactions and interests from rated items, establish the relationships between rated and unrated things, and find the latent satisfying and interesting items for users. In general, we propose two novel notions: **post-satisfaction** and **pre-interest**.

Definition 1. ***post-satisfaction****: for user u, and a rated item $i \in I_u^r$, if r_{ui} is larger than a threshold α_s, we call item i user u's post $-$ satisficaiton, $s_{ui} = 1$, $i \in I_u^{sr}$. Else, $s_{ui} = 0$. For user u, and an unrated item $i \in I_u^n$, we can get an evaluation of satisfaction s_{ui}^{eval} using the relations between rated and unrated items. If s_{ui}^{eval} is larger than a threshold α_s, we call item i user u's latent post $-$ satisficaiton, $s_{ui} = 1$, $i \in I_u^s - I_u^{sr}$. Else, $s_{ui} = 0$.*

Definition 2. ***pre-interest****: for user u, and a rated item $i \in I_u^r$, we call item i user u's pre $-$ interest, $t_{ui} = 1$, $i \in I_u^{tr}$. For user u, and an unrated item $i \in I_u^n$, like Definition 1, we can get an evaluation of interesting t_{ui}^{eval} using the relations between rated and unrated items. If t_{ui}^{eval} is larger than a threshold α_t, we call item i user u's latent pre $-$ interest, $t_{ui} = 1$, $i \in I_u^t - I_u^{tr}$. Else, $t_{ui} = 0$.*

Users' pre-interest means if a user u knows the existence of item i, there is a huge possibility that the u will use it or buy it, while users' post-satisfaction means i will get a high rating if u have used or bought it. The effect of these two notions for r_{ui} in Table I are shown in Table II.

Based on post-satisfaction and pre-interest, we define the $serendipity\ item$ in unrated items I_u^n:

Definition 3. ***serendipity item****: for user u, and an unrated item $i \in I_u^n$, we call item i user u's serendipity item, if and only if i is a post-satisfaction but not a pre-interest, which means i is relevant but unexpected, $i \in I_u^s - I_u^{sr}$, and $i \notin I_u^t - I_u^{tr}$.*

In Fig. 1, we can see the relationships between different categories of items in recommender system. Existing basic CF methods use only I_u^n as users' candidate items set (I_u^{can}). Some improved CF methods use additional information (users' property, items' property or some other contexts) to extract users' interest items (I_u^{can}). This kind of method can improve the accuracy,

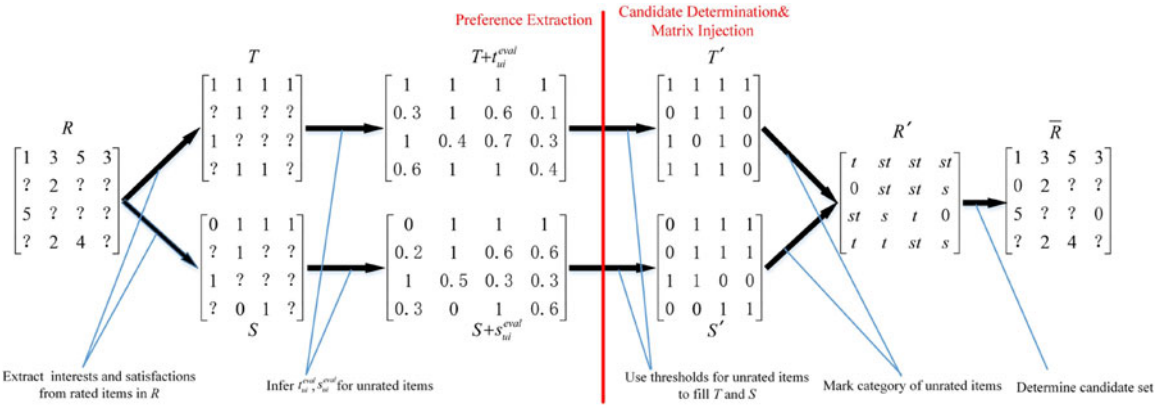


Fig. 2. Process of concise satisfaction and interest injection (CSII).

TABLE III
NOTATIONS IN THIS PAPER

Notation	Description
U	User set in our recommender system
I	Item set in our recommender system
r_{ui}	User u's ratings on item i
s_{ui}/t_{ui}	User u's satisfaction/interest flag of item i
$s_{ui}^{eval}/t_{ui}^{eval}$	User u's satisfaction/interest score of unrated item i
R	User-item matrix with ratings r_{ui}
S	A satisfaction matrix whose entry is s_{ui}
T	An interest matrix whose entry is t_{ui}
I_u^r	A set of rated items for user u
I_u^n	A set of unrated items for user u
I_u^s/I_u^t	A set of satisfying/interesting items for user u
I_u^{sr}/I_u^{tr}	A set of satisfying/interesting items for user u in I_u^r
I_u^{can}	A set of candidate items which will be evaluated for user u
I_u^{rec}	A set of recommended items for user u

but fails to offer users an opportunity to find serendipity item. Our proposed method uses post-satisfaction and pre-interest to tackle sparsity, ensure accuracy and offer serendipity. We define the core problem:

Problem Definition: *In recommender system, using I_u^r to predict post-satisfaction and pre-interest in I_u^n, and recommend Top-N items to users $\{k_1, k_2, k_3 ..., k_n\} \in I_u^{can}$, which satisfies the constraints:1)* $I_u^{rec} \in (I_u^t - I_u^{tr})$ or $(I_u^s - I_u^{sr})$, 2) $s_{u1}^{eval} \geq s_{u2}^{eval} \geq s_{u3}^{eval} ... \geq s_{un}^{eval} \geq \alpha_s$ and 3) $t_{u1}^{eval} \geq t_{u2}^{eval} \geq t_{u3}^{eval} ... \geq t_{un}^{eval} \geq \alpha_t$.

In general, the problems can be divided into 2 parts: 1) how to use post-satisfaction and pre-interest of I_u^r to extract users' preference in I_u^n, 2) how to verify I_u^{can} in I_u^n to tackle the user-item matrix R to improve the performance. The symbols used in our proposed approach are shown in Table III.

IV. PROPOSED APPROACH

Our proposed method, Concise Satisfaction and Interest Injection (CSII) is a two-stage approach:1) Preference Extraction, and 2) Candidate Set Determination&Matrix Injection. The overall process of our proposed method is shown in Fig. 2. In Stage 1, we build two basic matrices: post-satisfaction matrix S and pre-interest matrix T for rated items (I_u^r) in R (Step 1). Then we formulate the preference extraction problem for unrated items (I_u^n) into an improved one-class CF issue [30], and solve it with a novel ratio-Weighted Preference Matrix Factorization (rWPMF). We use threshold α_s, α_t to evaluate I_u^n, and build two full-value matrices S' and T' (Step 2). In Stage 2, we use S', T' to mark all unrated items (I_u^n) whether they are interesting, satisfying or serendipitous to build matrix R' (Step 3). Finally, we determine the candidate item set (I_u^{can}), set the ratings of $i \notin I_u^{can}$ to 0 and build a new user-item matrix $\overline{R}$ (Step 4).

After using our method to tackle original user-item matrix R, a variety of CF methods can use R' and $\overline{R}$ instead of R to ease the sparsity problem, ensure accuracy and improve the performance of serendipity in recommender system.

A. Stage 1: Preference Extraction

It is a core issue to infer users' preference accurately and efficiently in recommender system. We use only user-item matrix R to achieve preference extraction in order to reduce the complexity of computing.

1) Step 1. Preference Extraction for Rated Items: First, We extract users' preference from r_{ui} in R to build two matrices: post-satisfaction matrix S and pre-interest matrix T. We define post-satisfaction and pre-interest of I_u^r by ratings. We define all items in I_u^r as pre-interests, and parts of items with high ratings as post-satisfactions. The determination of s_{ui} in S and t_{ui} in T are shown in (1) and (2).

$$s_{ui} = \begin{cases} 1 & \forall i \in I_u^r, u \in U, r_{ui} \geq \alpha_s \\ 0 & \forall i \in I_u^r, u \in U, r_{ui} < \alpha_s \\ null & \forall i \in I_u^n \end{cases} \quad (1)$$

$$t_{ui} = \begin{cases} 1 & \forall i \in I_u^r, u \in U \\ null & \forall i \in I_u^n \end{cases} \quad (2)$$

As shown in Fig. 2, pre-interest matrix T contains $t_{ui} = 1$ for $i \in I_u^r$ to describe u's interests in i, and $t_{ui} = null$ for $i \in I_u^n$ is what we want to fill in *Step 2* of Preference Extraction.

While in post-satisfaction matrix S, it is very important to choose a proper threshold α_s to filter users' satisfaction for I_u^r. In our proposed method, we design three different ways to set α_s:

1) Average threshold of user: for every user u, and u's all rated items I_u^r, we can get α_s^u by (3):

$$\alpha_s^u = \sum_{i \in I_u^r}^{u} r_{ui} / |I_u^r| \tag{3}$$

2) Average threshold of item: for every item i, we can get α_s^i by (4) (N_i is the count of users who have rated i):

$$\alpha_s^i = \sum_{u \in U}^{i} r_{ui} / N_i \tag{4}$$

3) Relative-high threshold: because of the differences between each user and item, we define α_s^{rel} to be the top $\theta\%$ rating of user u. The three determinations of α_s can be applied for different applications. In Fig. 2, we choose α_s^u to give an example of our proposed method.

2) Step 2: Preference Extraction for Unrated Items: After building two matrices S and T, we need to get the latent post-satisfaction s_{ui}^{eval} and pre-interest t_{ui}^{eval} for i in I_u^n. From S and T, we find only three different values: 0,1 and $null$. We use 1 as positive effect while 0 as negative on latent preference of u. In T, we only have positive items, while in S, we have both positive and negative ones.

The methodology to infer latent preference of unrated items I_u^n is similar for post-satisfaction and pre-interest, but the details are different. According to the format of our problem, we consider employing One-Class CF (OCCF) to formulate our challenge [30]. Traditional OCCF problem can be applied to build latent pre-interest matrix $T + t_{ui}^{eval}$ directly [9]. But for latent post-satisfaction matrix $S + s_{ui}^{eval}$, which contains positive, negative and unknown examples, OCCF can not learn the training data well directly.

In our method, we formulate latent preference extraction problem into an improved one-class CF problem, and solve it with a novel ratio-Weighted Preference Matrix Factorization (rWPMF). OCCF uses only positive examples to mine the latent preference, and build a SVD-like Frobenius loss function. But in rWPMF, we use the same methodology as OCCF to build a **weighted** Frobenius loss function to build $T + t_{ui}^{eval}$ and $S + s_{ui}^{eval}$.

Given a matrix $T = (t_{ui})_{m \times n} \in \{1, null\}^{m \times n}$ and $S = (s_{ui})_{m \times n} \in \{0, 1, null\}^{m \times n}$, which contains m users and n items, and two corresponding weight matrices $W^t = (w_{ui}^t)_{m \times n} \in \mathbb{R}^{m \times n}$ and $W^s = (w_{ui}^s)_{m \times n} \in \mathbb{R}^{m \times n}$. Our proposed method rWPMF is aiming at approximating T and S with two low matrices $X^t = (x_{ui}^t)_{m \times n}$ and $X^s = (x_{ui}^s)_{m \times n}$ to minimize the objective of a weighted loss function as follows:

$$\mathcal{L}(X^t) = \sum_{u,i} w_{ui}^t (t_{ui} - x_{ui}^t)^2 \tag{5}$$

$$\mathcal{L}(X^s) = \sum_{u,i} w_{ui}^s (s_{ui} - x_{ui}^s)^2 \tag{6}$$

In the above loss functions (5) and (6), we employ common square loss which is usually applied in low-rank approximations. The weight matrices W^t and W^s show the importance to minimize the $\mathcal{L}$ for different examples in matrices S and T. In rWPMF, we use different weight-setting methods for pre-interest and post-satisfaction.

For pre-interest matrix T, we treat unknown ones as simple "negative" examples and set fixed weights δ_t for them: $w_{ui}^t = \delta; t_{ui} = null, u \in U, i \in I$ and set the weights for positive ones ($t_{ui} = 1$) as follows:

$$w_{ui}^t = \frac{|I_u^n|}{|I_u^r + I_u^n|} + \frac{N_i}{m}; t_{ui} = 1, u \in U, i \in I \tag{7}$$

In (7), $|\bullet|$ means the count of set $\bullet$. $|I_u^n| / |I_u^r + I_u^n|$ shows the fraction of unrated items I_u^n is positively correlated with the importance of rated items I_u^r. This idea is simple but reasonable. If one user rated lots of items, it seems that the user has a great interest in many kinds of items. The user's interests are wide and single item has low effect on the user's preference, so $|I_u^n| / |I_u^r + I_u^n|$ is small. But if the user rated only a few things, every single item the user has rated is very important for us to infer the user's preference, so $|I_u^n| / |I_u^r + I_u^n|$ is large. N_i/m means the popularity of item i, which is usually small because in real e-commence, $N_i \ll m$. Using (7) will probably increase accuracy because ratings for popular items are usually easier to predict and to find in the user profiles, while $|I_u^n| / |I_u^r + I_u^n|$ can ensure the personalization.

For post-satisfaction matrix S, which contains $0, 1, null$, we need to set weights for all three different examples: positive, negative and unknown. We tackle the unknown ones in S as what we do in pre-interest matrix T with δ_s. And set the weights for positive ones ($s_{ui} = 1$) and negative ones ($s_{ui} = 0$) as follows:

$$w_{ui}^s = \frac{|I_u^n|}{|I_u^r + I_u^n|} + \frac{|I_u^r - I_u^{sr}|}{|I_u^r|} + \frac{N_i}{m}; s_{ui} = 1, u \in U, i \in I \tag{8}$$

$$w_{ui}^s = \frac{|I_u^n|}{|I_u^r + I_u^n|} + \frac{|I_u^{sr}|}{|I_u^r|} + \frac{N_i}{m}; s_{ui} = 0, u \in U, i \in I \tag{9}$$

Following the same idea of weight setting in T, the first and third parts in (8) and (9) have the same meaning as described in (7). While $|I_u^r - I_u^{sr}|/|I_u^r|$ in (8) means the more user shows the post-satisfaction for rated items I_u^r, the less effect on post-satisfaction preference with one single satisfying item, and vice versa.

After building the weight matrix, we consider solving the optimization problem $argmin_X \mathcal{L}(X)$ efficiently and accurately. We use decomposition $X = PQ^T$ where $P \in \mathbb{R}^{m \times k}$ and $Q \in \mathbb{R}^{n \times k}$. Note that number of latent matrix P and Q's features, $k \ll r$ where r is the rank of the matrix S or T, $r = \min(m, n)$. With these process, we can rewrite the loss function (5) and (6) as follows:

$$\mathcal{L}(P^t, Q^t) = \sum_{u,i} \frac{w_{ui}^t}{2} (t_{ui} - P_{u.}^t (Q_{i.}^t)^T)^2 + \frac{\lambda^t}{2} (\Theta^t) \tag{10}$$

$$\mathcal{L}(P^s, Q^s) = \sum_{u,i} \frac{w_{ui}^s}{2} (s_{ui} - P_{u.}^s (Q_{i.}^s)^T)^2 + \frac{\lambda^s}{2} (\Theta^s) \tag{11}$$

In (10) and (11), $\frac{\lambda}{2}(\Theta)$ is the regularization term to prevent over-fitting. $\Theta = \|P_{u.}\|_F^2 + \|Q_{i.}\|_F^2, \|\bullet\|_F^2$ denotes the Frobenius norm and λ is the regularization parameter. We employ a

weighted alternating least squares [19] to solve low rankxbrk approximation problems. Use pre-interest matrix T as an example: We calculate partial derivatives of $\mathcal{L}$ for every entry of P^t, Q^t:

$$\begin{aligned}&\frac{\partial \mathcal{L}(P^t, Q^t)}{\partial P^t_{u.}}\\&= \left(\frac{\partial \mathcal{L}(P^t, Q^t)}{\partial P^t_{u1}}, ..., \frac{\partial \mathcal{L}(P^t, Q^t)}{\partial P^t_{ik}}\right)\\&= P^t_{u.}\left((Q^t)^T \overline{W^t_{u.}} Q^t + \lambda^t \left(\sum_i w^t_{ui}\right) I\right) - T_{u.}\overline{W^t_{u.}} Q^t\end{aligned}$$

Where $\overline{W^t_{u.}}$ is a diagonal matrix whose entry is the element of $W^t_{u.}$ on the diagonal, and I is a $k \times k$ identity matrix. Solve the partial derivative $\frac{\partial \mathcal{L}(P^t, Q^t)}{\partial P^t_{u.}} = 0$ with the fixed Q^t, we can get iterative formula (12) of P^t:

$$P^t_{u.} = T_{u.}\overline{W^t_{u.}} Q^t \left((Q^t)^T \overline{W^t_{u.}} Q^t + \lambda^t \left(\sum_i w^t_{ui}\right) I\right)^{-1} \tag{12}$$

Equivalently, with the fixed P^t, we can get iterative formula (13) of Q^t:

$$Q^t_{i.} = T_{.i}\overline{W^t_{.i}} P^t \left((P^t)^T \overline{W^t_{.i}} P^t + \lambda^t \left(\sum_u w^t_{ui}\right) I\right)^{-1} \tag{13}$$

Where $\overline{W^t_{.i}}$ is a diagonal matrix whose entry is the element of $W^t_{.i}$ on the diagonal.

Using these two iterative formulas (12) and (13), we can solve the problems with our proposed method rWPMF. Initialize P^t, Q^t with standard Gaussian random numbers (0 as mean and 0.01 as standard deviation) at first. Then update P^t, Q^t alternatively with (12) and (13). Repeat these iterative update procedures until convergence. Finally, we use $P^t(Q^t)^T$ to calculate every t^{eval}_{ui} to describe the latent preference of pre-interest for I^n_u.

For post-satisfaction, the procedure is the same as pre-interest. We summarize the above process in Algorithm 1 which we refer to as ratio-Weighted Preference Matrix Factorization (rWPMF).

Algorithm 1: Ratio-Weighted Preference Matrix Factorization (rWPMF).

Input: pre-interest matrix T, post-satisfaction matrix S, rank k
Output: Matrices $T + t^{eval}_{ui}, S + s^{eval}_{ui}$
1: Initialize P^t, Q^t, P^s, Q^s
2: Calculate W^t with (7)
3: Calculate W^s with (8), (9)
4: **repeat**
5: Update $P^t_{u.}, \forall u$ with (12)
6: Update $Q^t_{i.}, \forall i$ with (13)
7: **until** convergence
8: **repeat**
9: Update $P^s_{u.}, \forall u$ with (12)
10: Update $Q^s_{i.}, \forall i$ with (13)
11: **until** convergence
12: Calculate each $t^{eval}_{ui} = P^t_{u.}(Q^t_{i.})^T, s^{eval}_{ui} = P^s_{u.}(Q^s_{i.})^T$
13: Build $T + t^{eval}_{ui}, S + s^{eval}_{ui}$ with $T, S, t^{eval}_{ui}, s^{eval}_{ui}$
14: **return** Matrices $T + t^{eval}_{ui}, S + s^{eval}_{ui}$

B. Stage 2: Candidate Determination & Matrix Injection

In Stage 2, we use latent pre-interest t^{eval}_{ui} and post-satisfaction s^{eval}_{ui} to mark the unrated items I^n_u and build a marked matrix R'. Then we decide which item in I^n_u will be the candidate item. Finally, we make a Matrix Injection with candidates, building an intensified matrix $\overline{R}$. With R' and $\overline{R}$, we can achieve optimizations for recommender system.

1) Step 3. Candidate Determination: Using t^{eval}_{ui} and s^{eval}_{ui}, we can get latent preference for all unrated items in I^n_u. So we should use two thresholds to mark whether the user u likes the item i. α_s and α_t are applied in this process.

As proposed before, we employ (3) and (4) or relative-high threshold (in $Step$ 1) to calculate α_s for latent post-satisfaction s^{eval}_{ui}. While in our method, we use two ways to choose α_t: Fixed threshold α^{fix}_t or Average threshold of item α^i_t as follows:

$$\alpha^i_t = \frac{\sum_{i \in I^r_u} t_{ui} + \sum_{i \in I^n_u} t^{eval}_{ui}}{|U|} \tag{14}$$

α^i_t means the average pre-interest for item i in I. While in our example in Fig. 2, we choose $\alpha^{fix}_t = 0.5$ as pre-interest threshold and α^u_s (3) as post-satisfaction threshold to restrain the latent preference of users. T' and S' are constructed as follows:

$$t'_{ui} = \begin{cases} 0, & t^{eval}_{ui} < \alpha_t \\ 1, & t^{eval}_{ui} > \alpha_t, \text{or}, t_{ui} = 1 \end{cases} \tag{15}$$

$$s'_{ui} = \begin{cases} 0, & s^{eval}_{ui} < \alpha_s, \text{or}, s_{ui} = 0 \\ 1, & s^{eval}_{ui} > \alpha_s, \text{or}, s_{ui} = 1 \end{cases} \tag{16}$$

With these t'_{ui}, s'_{ui}, we can build a new matrix: marked matrix R'. r'_{ui} in R' is consisted of two different categories of factors: interest mark t, satisfaction mark s. R' are constructed as follows:

$$r'_{ui} = \begin{cases} st, s'_{ui} = 1, t'_{ui} = 1 \\ s, s'_{ui} = 1, t'_{ui} = 0 \\ t, s'_{ui} = 0, t'_{ui} = 1 \\ 0, s'_{ui} = 0, t'_{ui} = 0 \end{cases} \tag{17}$$

In (17), we can finally find the candidate item set in accordance with r'_{ui}. Candidate item set I^{can}_u will be the pre-interest items ($r'_{ui} = t$), post-satisfaction items ($r'_{ui} = s$) either or both of them ($r'_{ui} = st$) in unrated items (I^n_u). The reason why we also mark the rated items (I^r_u) in R' is that we should use the users' preference of interest or satisfaction to make a customized recommendation, which will be discussed in the following subsection.

2) Step 4: Matrix Injection: We want to utilize the attribute of I^{can}_u to tackle the original user-item matrix R, and build an intensified user-item matrix $\overline{R}$. $\overline{R}$ will be applied in many fashionable CF recommender methods instead of original

user-item matrix R to improve the performance of CF. We employ a Matrix Injection method to achieve improvement. In original matrix R, there are billions of items that users haven't rated. In these items (I_u^n), users may only be interested in or satisfied with a few items in accordance with preferences extracted from users' rated items (I_u^r). But the difference between existing methods and CSII is that we should mark the uninteresting and unsatisfying items in $\overline{R}$, instead of using the interesting and satisfying ones. So $\overline{r}_{ui}$ in $\overline{R}$ is as follows:

$$\overline{r}_{ui} = \begin{cases} r_{ui}, & i \in I_u^r \\ 0, & r'_{ui} = 0, i \in I_u^n \\ null, & else \end{cases} \tag{18}$$

Using Matrix Injection, we filter uninteresting and unsatisfying items in original user-item matrix R. And using R', $\overline{R}$, we can present a better performance when we apply CSII before traditional CF methods. Firstly, CSII uses original user-item matrix R to get R', $\overline{R}$. We run any fashionable CF methods (MF-based, User-based or item-based) on intensified user-matrix $\overline{R}$, and with the results of this CF method, we proposed a customized recommendation to recommend the properest item to the user with marked user-item R'.

The customized recommendation uses users' preference of interests and satisfactions to decide which item should be recommended first in the recommender list. For example, a user u has very high pre-interest and low post-satisfaction (In R', number of $r'_{ui} = t$ is larger than $r'_{ui} = s$), we can conduct that pre-interest has great effect on this user, so we can arrange the recommender list in accordance with pre-interest rank. The customized recommendation also can use post-satisfaction rank to add the serendipity, or both of them to ensure the accuracy of CF.

The complete CSII method is described in Algorithm 2:

Algorithm 2: Concise Satisfaction and Interest Injection (CSII).

Input: original user-item matrix R
Output: personalized recommender list L
Strategy 1:Preference Extraction
1: Initialize pre-interest matrix T, post-satisfaction matrix S, threshold α_t, α_s
2: Calculate $T + t_{ui}^{eval}, S + s_{ui}^{eval}$(**Algorithm 1:rWPMF**)
Strategy 2:Candidate Determination&Matrix Injection
3: Use $T + t_{ui}^{eval}, S + s_{ui}^{eval}$ to build matrix T' and S'
4: Mark items in I to determine candidate item set I_u^{can}
5: Use T' and S' to build marked matrix R'
6: Use Matrix Injection to build intensified user-item matrix $\overline{R}$
Recommender strategy
7: Apply some CF method on $\overline{R}$ instead of R
8: Get original recommender list L'
9: Apply marked matrix R' on L' to build personalized recommender list L
10: **return** personalized recommender list L

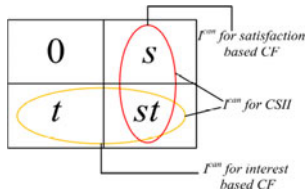


Fig. 3. Candidate item set decision for different CF methods.

V. Optimal Performance Analysis

In this section, we want to describe two optimal issues of CSII: 1) why we mark uninteresting and unsatisfying items in intensified matrix $\overline{R}$ instead of interesting and satisfying ones. 2) how CSII can improve the performance of CF.

The reason why we fill the missing value with 0 to mark uninteresting and unsatisfying items is 1) the importance of pre-interest and post-satisfaction for unrated items are unclear. For some users, pre-interest may affect the preference much more than post-satisfaction, while the other users are inverse. But it is true that both factors will act in users' preference. If we fill the interesting and satisfying items to build a new matrix, we must decide which factor is more important for each user, which is unreasonable and increasing the computing complexity. 2) By this method, we ensure that all the possible items can be chosen by users, which will be reserved in candidate item set instead of being ignored mistakenly. Existing improved CF methods can be divided by the way they decide candidate item set, as shown in Fig. 3. Satisfaction-based method chooses users' post-satisfaction item as I^{can}, and interest-based choose pre-interest ones. But CSII uses both of them to make candidate item set accurate and comprehensive. Our method can improve CF methods from two aspects: 1) By preference extraction in CSII, we can find users' preferences only by their ratings without additional information, which saves computing resources. Utilizing pre-interest and post-satisfaction, users' preference can instruct the recommender system to make a customized recommendation. 2) By candidate determination and matrix injection, we can ease data sparse and cold-start issues and improve the performance of existing CF methods. By injecting 0 to intensified matrix $\overline{R}$, we can ease data sparse and cold-start issues in the utmost extent, which is quite obvious. To present the ground for how to improve the performance of CF, we try to apply CSII on two most popular CF methods: Item-based method (IM) [17] and MF-based method (MF) [18].

In IM method, we predict a rating r_{ui}^{im} for a target item i of a user u by referencing his ratings on those items similar to item i. As shown in Fig. 4, we want to predict r_{23}^{im}. In basic IM, we use original user-item matrix R, and u_2's rated item i_2 to make the prediction. Because u_2's rated item set is too small (always happen in real recommender system), basic IM cannot get accurate and efficient results. However, if we use CSII to build intensified matrix $\overline{R}$, apply IM on $\overline{R}$ instead of R, we can compute the similarity of two items: i_1 and i_2, and utilize more ratings in rated item set, which can greatly improve the accuracy of recommender results. Meanwhile, with the matrix injection, we find that similarity between two different items can be calculated more accurately because every item has more

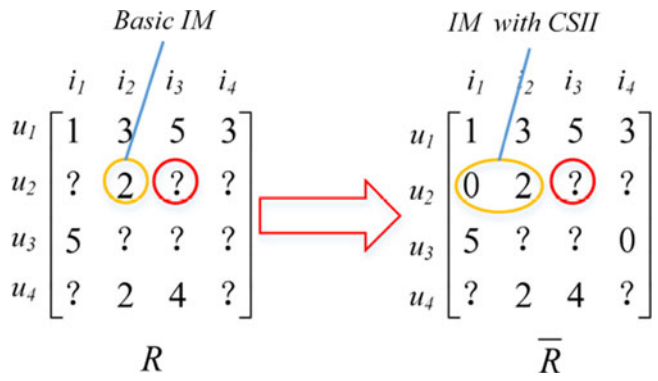


Fig. 4. How CSII improves basic item-based CF.

"ratings" in $\overline{R}$ (ratings, uninteresting and satisfying mark 0). In MF-based method, if is given an original user-item matrix R, MF factorizes it into the inner product of two low rank matrices X, Y with a dimension f. X, Y are the latent feature matrices for users ($X \in \mathbb{R}^{m \times f}$) and items ($Y \in \mathbb{R}^{f \times n}$). And every prediction of items for users can be calculated: $r_{ui}^{mf} = X_{u.} Y_{.i}$. MF method tries to find the latent relations between different users and items. But with R in Fig. 4, MF cannot understand u_2 has a relationship with u_3 because there are no common ratings in u_2's rated item set and u_3's rated item set. However, with CSII applied and using the intensified matrix $\overline{R}$, we can realize the relationships successfully because they have a common "rating" on i_1. Also, MF method can find i_4 has a stronger relationship with i_1 than i_2 and i_3 because they are "rated" by both u_1 and u_3 in $\overline{R}$, which may be missed in original matrix R. Therefore, MF can factorize better latent matrices representing more latent relationships among users and items with CSII, which improves the overall accuracy and efficiency.

Otherwise, we can make customized recommender results for each user based on marked matrix R' (Fig. 2). If we want to achieve some special recommender results, we can re-rank the Top-N list based on the mark r'_{ui} in R'. For every rated or unrated item i for user u, we have r'_{ui} to describe users' preference. The amount of different r'_{ui} (st, s, t) means users' preference for accuracy, post-satisfaction and pre-interest. For example, if we want to achieve an accurate Top-N result, we can arrange the list as the rank $r'_{ui} = st \geq t \geq s \geq 0$, while if we want serendipitous ones, the list will be ranked as $r'_{ui} = s \geq st \geq t \geq 0$. Customized recommendation can ensure the accuracy for each user and enrich the serendipity and diversity of recommender system.

VI. Experiment

In this section, we will discuss the accuracy, efficiency, and serendipity of our proposed method CSII with a real-life dataset. We first introduce the data set and the evaluation indicators we employed in our experiment. Then we verify our approach from: 1) Accuracy of Preference Extraction, 2) Effect of threshold α, 3) Improvement of CF methods with CSII, 4) Serendipitous and customized Top-N recommendation and 5) Complex Analysis.

TABLE IV
MovieLens Datasets Description

	ML100k	ML1M	ML10M	Yelp
m(users)	943	6,040	71,567	43,873
n(items)	1682	3,900	10,681	11,537
r(ratings)	100,000	1,000,209	10,000,054	229,907
sparsity	6.79%	4.24%	1.30%	0.43%

A. Dataset Description and Evaluation Indicators

We apply experiments with Movielens datasets (Movielens 100k, 1M, 10M) and Yelp for RecSys. Especially, Movenlens 100k has 100,000 ratings for 1682 items by 943 users with different IDs. Ratings range from 1 to 5, and each user has 20 ratings as least. The sparsity of this dataset is 6.79% (calculated as: $\frac{|I^r|}{|m \times n|}$). Lenskit [31] is implemented to build the experiment environment. All statistical data of datasets we used are shown in Table IV.

To measure the accuracy of preference extraction, we use err, which is calculated as: $err = 1 - \frac{|I^{ex} \cap I^{real}|}{|I^{real}|}$. I^{ex} means the item set we extracted from unrated item set I^n as candidate item set, and I^{real} means the item set users are really interested in or satisfied with (have rated in test set). To measure the performance of Top-N recommendation, we use precision ($P@N = \frac{|I^{rec} \cap I^{real}|}{|I^{rec}|}$) and recall ($R@N = \frac{|I^{rec} \cap I^{real}|}{|I^{real}|}$), where I^{rec} is the Top-N recommendation item set. And we use $normalized\ discounted\ cumulative\ gain$ (nDCG) as the indicator for Top-N recommendation, which is usually employed in recommender system. In order to prove the improvement of basic CF methods, we employ Item-based CF method (IM) [17] and MF-based CF method (MF) [18] as our baseline methods.

Otherwise, we consider the serendipity of Top-N recommendation [2], [15]. We use (19) ([16]) to measure the serendipity. I^{real}, I^{rec}, I^s and I^t are users' real rated items, recommendation items, satisfactory items and interesting items, as described before. To achieve the best evaluation of our method, we use 70% as training dataset and the other as test set. All the measurements are averaged using 5-crossed validation.

$$Serep = \frac{\left|I^{rec} \cap I^{real} \cap (I^s - I^t)\right|}{|I^{real}|} \tag{19}$$

B. Experiment and Discussion

We choose several baselines to test our method in the whole dataset: interest-based method (ITBM) [9], satisfaction-based method (SBM), item-based method (IBM) [17], popular method (PM), random method (RM) and two state-of-the-art methods element-ALS [32] and BPR [33]. ITBM borrows the idea of OCCF, and calculate users' interest for each unrated items, while SBM only infer users' satisfaction for unrated items. IBM produces the preference scores exploiting the item-based CF, and using the original user-item matrix R. RM is a basic baseline method that does not extract preference but randomly selects items among unrated items, while PM recommends the most popular ones to users. eALS and BPR are usually applied in

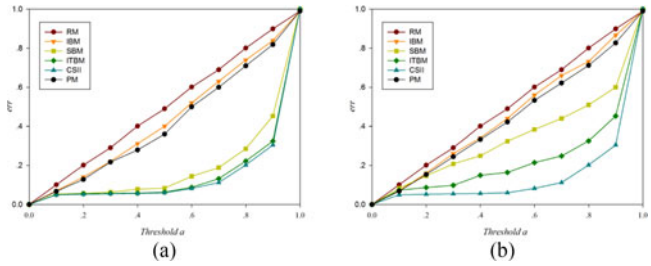


Fig. 5. Accuracy of preference extraction on M100K and Yelp. (a) M100k. (b) Yelp.

implicit feedback recommender system, and they also can be used on the dataset of ratings as explicit feedback ones.

With no special introduction, All CF method should use 5 nearest neighbours to do the predictions. And MF-based method should depose R (m users and n items) into P ($m \times k$) and Q ($k \times n$) with $k = Rank(R)$. Learning rate is 0.05. As our method is an improvement of Top-N recommendation with CF methods, we vary N from 5 to 20 with an increase step of 5.

1) Accuracy of Preference Extraction: To make this experiment simple and direct, we use the same fixed threshold α_s and α_t in ITBM, SBM, and CSII, which is represented by $\alpha = 0.8$. We also employ the same parameter settings for all methods which employ OCCF method.

Fig. 5 shows the accuracy of preference extraction with different methods on M100k and Yelp. It is obvious that RM leads to the worst performance on preference extraction. PM and IBM are relatively better methods than RM, but still not good enough to compare with the three knowledge-based method: ITBM, SBM and CSII. ITBM and SBM use users' interest and satisfaction respectively while CSII combines them. ITBM ignores the item user may not be interested in but satisfied with after using (the serendipity item we describe before) and SBM ignores the interesting items, which are the most useful items we should consider. It is obvious that our method can take a more stable performance than SBM and ITBM, especially when the dataset is more sparse (Yelp).

Obviously, lower α can get better err, but the ability to filter the candidate is also weak, which will be approved in $ComplexAnalysis$. And we should trade off the err and the filter ability. In Fig. 5, we can find err changed sharply between $\alpha = 0.85$ and 0.95. So we decide to find which α could achieve the best performance.

2) Effect of Threshold α: We employ two CF methods: Item-based method (IM) [17] and MF-based method (MF) [18], with different threshold α in CSII with M100K dataset to verify the effect of threshold. We choose precision, recall as the performance indicators.

As shown in Fig. 5, if we fix single threshold α_s or α_t and change the other, preference extraction of CSII will be simplified into ITBM or SBM. So we change α_s and α_t together as α, and apply the threshold α in two CF methods to measure the effects. We increase α in an increment of 0.1 for the range of 0.0–1.0 at first [Fig. 6(a) and (b)], and increase α in an increment of 0.01 for the best performance range 0.83–0.93 for item-based CF [Fig. 6(c)] and 0.78–8.88 for MF-based CF [Fig. 6(d)]. In the

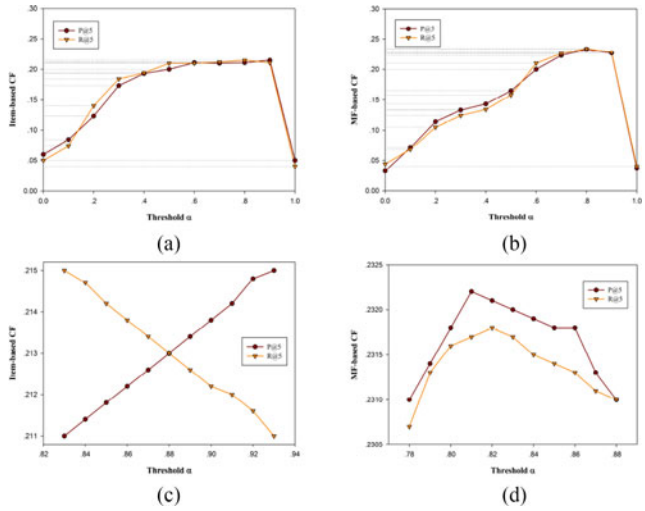


Fig. 6. Effect of threshold α on different CF methods. (a) α (0.0–1.0) on item-based CF. (b) α (0.0–1.0) on MF-based CF. (c) α (0.83–0.93) on item-based CF. (d) α (0.78–0.88) on MF-based CF.

first two results, we can find when α is small, the performance of both CF methods is not good enough, although err of preference extraction in Fig. 5 is better. The reason is that, with lower threshold, the candidate set is huge relatively close to the basic method, and CSII cannot pick up the most possible candidate items accurately and efficiently with them.

To make our verification better, in Fig. 6(c) and (d), we magnify the best performance range in different methods. In our experiment, we get the best ones when $\alpha = 0.88$ in item-based CF and $\alpha = 0.81$ in MF-based CF. And the best performance of these two methods is much better than basic ones ($\alpha = 0$ or 1). The improvement of CF methods with CSII compared with basic CF methods will be discussed in next section.

3) Improvement of CF Methods with CSII: The most important contribution of our work is to apply CSII on basic CF methods and improve the performance. In this section, we choose three baseline improved-methods: ITBM [9], SBM, and our proposed method CSII. We apply them with two basic CF methods: Item-based CF [17] and MF-based CF [18]. We use M100K as the dateset, employing three different performance indicators: Precision, Recall and nDCG, and recommend 5, 10, 15, 20 items to each user.

Table V shows the improvement of all CF methods with different improved methods. $basic$ means CF methods without any improved method. The numbers in boldface indicate the highest accuracy among all CF methods.

Among existing $basic$ CF methods, MF-based CF brings about the best performance while Item-based CF brings the worst. In literature, MF-based methods (SVD, SVD++ and pureSVD) are known to provide better performance than Item-based methods. We observe that ITBM, SBM and our proposed CSII can improve the performance of all basic CF methods. For example, these methods improve Precision@10 of Item-based CF, MF-based CF by about 4, 3, and 5 times and 3, 1.5, 4 times, respectively. We see less improvement in performance when our method is applied to MF-based CF than to Item-based CF. The

TABLE V
IMPROVEMENT OF CF METHODS WITH DIFFERENT METHODS ($\alpha = 0.85$)

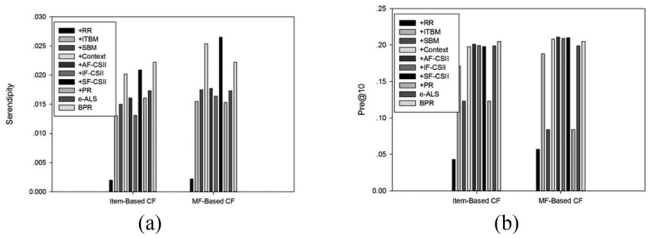
	Metric	Item-based CF				MF-based CF			
		basic	+ITBM	+SBM	+CSII	basic	+ITBM	+SBM	+CSII
Precision	@5	0.039	0.191	0.133	**0.199**	0.067	0.190	0.097	**0.202**
	@10	0.043	0.171	0.123	**0.203**	0.057	0.188	0.084	**0.211**
	@15	0.038	0.162	0.110	**0.193**	0.053	0.179	0.067	**0.188**
	@20	0.042	0.131	0.098	**0.193**	0.049	0.164	0.078	**0.213**
Recall	@5	0.027	0.192	0.142	**0.223**	0.063	0.201	0.102	**0.233**
	@10	0.065	0.311	0.132	**0.363**	0.093	0.213	0.097	**0.313**
	@15	0.073	0.345	0.145	**0.423**	0.141	0.223	0.156	**0.253**
	@20	0.088	0.411	0.193	**0.493**	0.153	0.232	0.172	**0.343**
nDCG	@5	0.037	0.193	0.088	**0.233**	0.077	0.262	0.110	**0.353**
	@10	0.054	0.198	0.121	**0.216**	0.084	0.222	0.099	**0.378**
	@15	0.045	0.201	0.142	**0.229**	0.088	0.245	0.102	**0.393**
	@20	0.062	0.211	0.183	**0.253**	0.095	0.298	0.123	**0.433**

reason is that MF-based method treats all missing ratings as 0 in user-item matrix. However, it cannot find out the possible interesting and satisfying unrated items as CSII can do. Furthermore, our method can find the uninteresting and unsatisfying items to improve the performance of MF-based CF.

Among all CF methods with improvement, CF with CSII shows the best performance, followed by methods with ITBM. Our method improves both basic CF methods greatly (average 2–5 times) because they consider all unrated items as unknown ones. Using CSII, they can extract users' preference correctly by regarding zero ratings as uninteresting and unsatisfying ones and null values as unknown ones. We also note that although our method is the best one among ITBM, SBM and basic, the improvement is not so remarkable compared to CSII and ITBM. The reason is simple: as shown in Fig. 3, ITBM chooses users' latent interesting items as candidate while CSII chooses intereseting or satisfying items. The item which is satisfying but uninteresting is called $serendipitous\ items$, which CSII chooses but ITBM does not. In recommender system, the number of users' interesting items is much more than serendipitous items'. So the improvement of CSII compared with ITBM is not so much as compared with SBM and basic, though it has been improved more than 10% already.

4) Serendipitous and Customized Top-N Recommendation: We compare different methods on two basic CF methods: Random Rank (RR), Popularity Rank (PR), ITBM [9], SBM, Context-based method [2], AF-CSII, IF-CSII, SF-CSII and two state-of-the-art methods: element-ALS [32] and BPR [33]. To verify the advantage of our method, we employ a context-based CF [2] to make serendipitous recommendations with additional context information, which includes the attributes of items, users and ratings. We use M100k as dataset and learning rate is 0.05, same for all the methods.

As mentioned in Section IV, CSII can make different recommender ranks after CF methods. Without loss of generality, we make three recommender rank of CSII: Accuracy First CSII (rank the recommender list as $r'_{ui} = st \geq t \geq s \geq 0$), Interest First CSII ($r'_{ui} = t \geq st \geq s \geq 0$) and Serendipity First CSII (also satisfaction first, $r'_{ui} = s \geq st \geq t \geq 0$). All the parameters (like α) are fixed in this experiment. We evaluate serendipity with (19).



(a) (b)

Fig. 7. Serendipity and accuracy performance. (a) Serendipity. (b) Precision@10.

From Fig. 7(a), we can see RR gives the worst performance in serendipity, while SF-CSII performs best. Note that context-based method, eALS and BPR are the efficient and popular ones to mine the serendipity items in unrated items, but our proposed method can make a recommendation as context-based method does, and even a little better in serendipity issues. In Fig. 7(b), we find some methods perform better in serendipity, but not so good in accuracy (ITBM, SBM, PR). And the performance that context-based method can achieve in accuracy and serendipity is very close to our proposed method. However, CSII uses only ratings to make recommendation, which reduces the cost of storage and computing. This experiment shows that CSII can maintain a relative high level on accuracy with a more serendipitous recommendation without additional information, while other methods usually fail to do that. Meanwhile, with different segments (Accuracy-First, Interest-First and Serendipity-First), CSII can be applied into any special recommender system which needs customized recommendations.

5) Complex Analysis: For space analysis, because we add only two more matrices. $\overline{R}$ and R' are both in $\mathbb{R}^{m \times n}$. So if we apply our proposed method on any existing method, it will cost additional $O(2 \times m \times n)$ storage. Compared with the ability of computing and storing of distributed systems (like Hadoop), it is not a great space cost. However, it can improve the performance greatly (as shown in our experiments before) and reduce recommend time by solving the data sparse and cold-start.

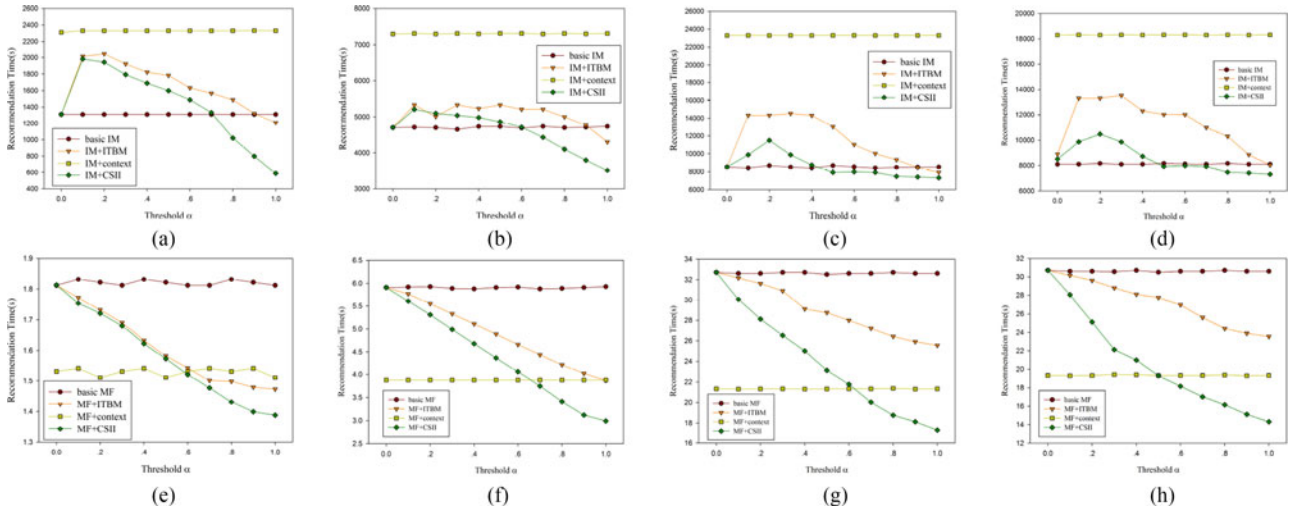


Fig. 8. Recommendation time with different methods. (a) IM methods on M100K. (b) IM methods on M1M. (c) IM methods on M10M. (d) IM methods on Yelp. (e) MF methods on M100K. (f) MF methods on M1M. (g) MF methods on M10M. (h) MF methods on Yelp.

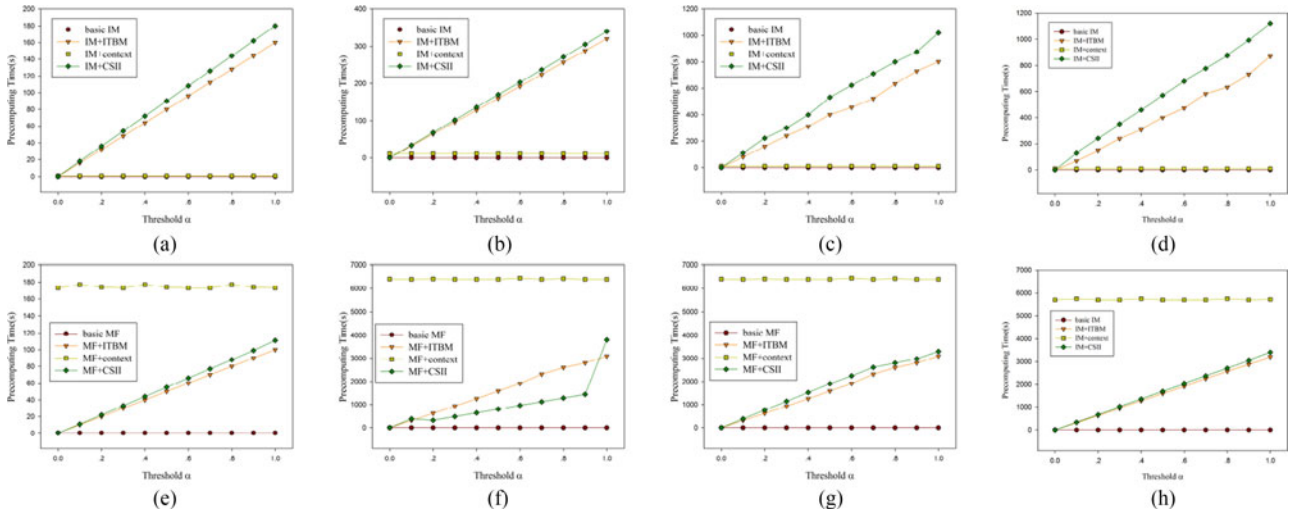


Fig. 9. Precomputing Time with different methods. (a) IM methods on M100K. (b) IM methods on M1M. (c) IM methods on M10M. (d) IM methods on Yelp. (e) MF methods on M100K. (f) MF methods on M1M. (g) MF methods on M10M. (h) MF methods on Yelp.

For time analysis, our method has the same time complexity as the CF which employed our method, with additional time cost $O(m \times n \times n)$ in pre-computing time to compute $\overline{R}$ and R'. But it can reduce the recommendation time by enhancing the basic rating matrix. So we use recommendation time, pre-computing time as the performance indicator. We compare the running time of two basic CF methods with and without our approach. CSII has both strength and weakness in terms of execution times. Note that the weakness is shown at the pre-computation, while the strength happens at the recommendation.

CSII can reduce recommendation time because it significantly reduces the number of candidate items whose relative preferences need to be predicted accurately and efficiently, and ease the sparsity issues. Meanwhile, our approach may require a little more pre-computation time because it has to do preference extraction. We study the trade-off of our approach by examining the running time when CSII is applied to both Item-based CF and MF-based CF, with three data sets of different quantity (M100k, M1M, M10M) and Yelp. As shown in Figs. 8 and 9, we use basic CF, ITBM [9], context-based method [2] as our baseline methods. We find that basic IM and context-based methods have a stable performance in recommendation time. Basic MF has more candidate items than any other method, the recommendation time of basic method is the longest of all methods. Note that ITBM, context-based method and CSII have the ability to filter the candidate items before recommendation, so the recommendation time is short relatively.

We focus on ITBM and our proposed CSII, which have similar performance in Precision, Recall, nDCG and recommendation time. In IM, CSII can reduce the recommendation time more than average 20% when α is set between 0.7–1.0 in M100k, 0.6–1.0 in M1M and 10% between 0.4-1.0 in M10M compared with ITBM. The more data we use, the better performance CSII can give in recommendation time. The reason is that CSII can find three different candidate items in unrated items $(r' = st, t, s)$, while ITBM only find two $(r' = st, t)$. When

dataset becomes huge enough, there are many satisfaction items which ITBM may ignore but CSII does not. We can notice that with dataset growing, CSII can reach its best performance more rapidly than ITBM. CSII uses two matrices (pre-interest and post-satisfaction) to extract preference while ITBM uses only one (pre-interest). With more accurate preference extraction, CSII outperforms ITBM in recommendation time in IM methods with different scales of datasets. When it comes to MF method, ITBM and CSII can both reduce recommendation time to a great extent. However, because of the same reason in item-based method, our proposed method CSII outperforms ITBM in MF methods.

In Fig. 9, we consider pre-computing time as the performance indicator. In item-based methods (IM), basic method and context-based method need no precomuting time because all the computations are done in recommendation process. But in MF methods, context-based method needs to compute the relationship between different users and items, which takes quite a lot of time. That's an advantage of CSII, because it can achieve the same performance (shown in Fig. 7) as context-based method but with less additional information.

Note that in Fig. 9, ITBM needs less pre-computing time than CSII in IM, CF methods in all three datasets. The reason is that ITBM only bulids one model to help recommendation, while CSII should build two models with OCCF methods. However, if we add up precomputing time (Fig. 9) and recommendation time (Fig. 8), we can see that when considering the aggregate time of recommendation, CSII is still better than ITBM, needless to say its out-performance in preference extraction, accuracy and serendipity.

Our approach can reduce the recommendation time of IM and MF in different scales of datasets while needing more pre-computation time. Note that recommendation time is more crucial than pre-computation time in recommender system, we believe that CSII improves those CF methods in terms of running time. In summary, our method can improve existing CF methods and find serendipity items accurately and carefully with acceptable time and space cost.

VII. Conclusion

While many existing CF methods mainly focus on the rated items and the context of users and items to improve recommendation performance, we try to use only user-item rating matrix without any additional information to make a recommendation accurately, efficiently and serendipitously. Based on this motivation, we proposed a novel method, Concise Satisfaction and Interest Injection (CSII), to tackle the unrated items and find the candidate item set, which includes users' latent interesting items, latent satisfying items and serendipitous items. CSII employs One-Class CF (OCCF) to extract users' pre-interesting and post-satisfaction items, and build a preference matrix for unrated items. However, CSII does not pick up the candidate items, but filter the unsatisfying and uninteresting items with 0 to build a new rating matirx (intensified matrix). With preference matrix and intensified matrix, traditional CF methods can make a better recommendation as CSII applied on them. Through comprehensive experiments on different scales of datasets, we demonstrated that CSII is effective and practical, dramatically improving the accuracies, efficiencies and serendipities of existing CF methods (Item-based CF and MF-based CF) without additional information.

References

[1] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender system application developments," *Decision Support Syst.*, vol. 74, pp. 12–32, 2015.

[2] X. Lei, X. Qian, and G. Zhao, "Rating prediction based on social sentiment from textual reviews," *IEEE Trans. Multimedia*, vol. 18, no. 9, pp. 1910–1921, Sep. 2016.

[3] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems : A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowledge Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[4] G. Zhao, X. Qian, and X. Xie, "User-service rating prediction by exploring social users' rating behaviors," *IEEE Trans. Multimedia*, vol. 18, no. 3, pp. 496–506, Mar. 2016.

[5] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutirrez, "Recommender systems survey," *Knowledge-Based Syst.*, vol. 46, no. 1, pp. 109–132, 2013.

[6] Z. Yu, M. Tian, Z. Wang, B. Guo, and T. Mei, "Shop-type recommendation leveraging the data from social media and location-based services," *ACM Trans. Knowledge Discovery Data*, vol. 11, no. 1, pp. 1–21, 2016.

[7] J. B. Schafer, F. Dan, J. Herlocker, and S. Sen, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, 2004.

[8] G. Adomavicius and J. Zhang, "Improving stability of recommender systems: A meta-algorithmic approach," *IEEE Trans. Knowledge Data Eng.*, vol. 27, no. 6, pp. 1573–1587, Jun. 2015.

[9] W. S. Hwang, J. Parc, S. W. Kim, J. Lee, and D. Lee, "Told you i didn't like it: Exploiting uninteresting items for effective collaborative filtering," in *Proc. IEEE Int. Conf. Data Eng.*, 2016, pp. 349–360.

[10] W. X. Zhao *et al.*, "Connecting social media to e-commerce: Cold-start product recommendation using microblogging information," *IEEE Trans. Knowledge Data Eng.*, vol. 28, no. 5, pp. 1147–1159, 2016.

[11] G. Zhao, X. Qian, X. Lei, and T. Mei, "Service quality evaluation by exploring social users contextual information," *IEEE Trans. Knowledge Data Eng.*, vol. 28, no. 12, pp. 3382–3394, Dec. 2016.

[12] Z. Zhao, H. Lu, D. Cai, X. He, and Y. Zhuang, "User preference learning for online social recommendation," *IEEE Trans. Knowledge Data Eng.*, vol. 28, no. 9, pp. 2522–2534, Sep. 2016.

[13] M. Sarwat, J. J. Levandoski, A. Eldawy, and M. F. Mokbel, "Lars*: An efficient and scalable location-aware recommender system," *IEEE Trans. Knowledge Data Eng.*, vol. 26, no. 6, pp. 1384–1399, Jun. 2014.

[14] M. Jiang *et al.*, "Social recommendation with cross-domain transferable knowledge," *IEEE Trans. Knowledge Data Eng.*, vol. 27, no. 11, pp. 3084–3097, Nov. 2015.

[15] M. Ge, C. Delgado-Battenfeld, and D. Jannach, "Beyond accuracy: Evaluating recommender systems by coverage and serendipity," in *Proc. ACM Conf. Recommender Syst. Recsys*, Barcelona, Spain, Sep. 2010, pp. 257–260.

[16] D. Kotkov, S. Wang, and J. Veijalainen, "A survey of serendipity in recommender systems," *Knowledge-Based Syst.*, vol. 111, pp. 180–192, 2016.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. Int. Conf. World Wide Web*, 2001, pp. 285–295.

[18] S. Zhang, W. Wang, J. Ford, and F. Makedon, "Using singular value decomposition approximation for collaborative filtering," in *Proc. IEEE Int. Conf. E-Commerce Technol.*, 2005, pp. 257–264.

[19] K. Gabriel and S. Zamir, "Lower rank approximation of matrices by least squares with any choice of weights," *Technometrics*, vol. 21, no. 4, pp. 489–498, 1979.

[20] X. Qian, H. Feng, G. Zhao, and T. Mei, "Personalized recommendation combining user interest and social circle," *IEEE Trans. Knowledge Data Eng.*, vol. 26, no. 7, pp. 1763–1777, Jul. 2014.

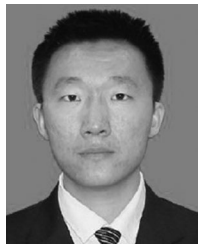
[21] G. Zhao, X. Qian, and C. Kang, "Service rating prediction by exploring social mobile users geographical locations," *IEEE Trans. Big Data*, vol. 3, no. 1, pp. 67–78, Mar. 2017.

[22] L. Sun, X. Wang, Z. Wang, H. V. Zhao, and W. Zhu, "Social-aware video recommendation for online social groups," *IEEE Trans. Multimedia*, vol. 19, no. 3, pp. 609–618, Mar. 2016.

[23] T. Han *et al.*, "Dancelets mining for video recommendation based on dance styles," *IEEE Trans. Multimedia*, vol. 19, no. 4, pp. 712–724, Apr. 2016.
[24] Z. Yu, H. Xu, Z. Yang, and B. Guo, "Personalized travel package with multi-point-of-interest recommendation based on crowdsourced user footprints," *IEEE Trans. Human-Mach. Syst.*, vol. 46, no. 1, pp. 151–158, Feb. 2016.
[25] L. Hu, A. Sun, and Y. Liu, "Your neighbors affect your ratings: On geographical neighborhood influence to rating prediction," in *Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2014, pp. 345–354.
[26] S. Jiang, X. Qian, T. Mei, and Y. Fu, "Personalized travel sequence recommendation on multi-source big social media," *IEEE Trans. Big Data*, vol. 2, no. 1, pp. 43–56, Mar. 2016.
[27] P. Lou, G. Zhao, X. Qian, H. Wang, and X. Hou, "Schedule a rich sentimental travel via sentimental poi mining and recommendation," in *Proc. IEEE 2nd Int. Conf. Multimedia Big Data*, 2016, pp. 33–40.
[28] X. He and T. S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proc. 40th Int. ACM SIGIR Conf. Res. & Develop. Inf. Retrieval*, 2017, pp. 355–364.
[29] X. He *et al.*, "Neural collaborative filtering," in *Proc. Int. Conf. World Wide Web*, 2017, pp. 173–182.
[30] R. Pan, Y. Zhou, B. Cao, and N. N. Liu, "One-class collaborative filtering," vol. 29, no. 5, pp. 502–511, 2008.
[31] M. D. Ekstrand, M. Ludwig, J. A. Konstan, and J. T. Riedl, "Rethinking the recommender research ecosystem: Reproducibility, openness, and lenskit," in *Proc. ACM Conf. Recommender Syst. Recsys*, Chicago, IL, USA, Oct. 2011, pp. 133–140.
[32] X. He, H. Zhang, M. Y. Kan, and T. S. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *Proc. Int. ACM SIGIR Conf. Res. Dev. Inf. Retrieval*, 2016, pp. 549–558.
[33] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proc. 25th Conf. Uncertainty Artif. Intell.*, 2012, pp. 452–461.

Yongjian Yang received the B.E. degree in automatization from the Jilin University of Technology, Changchun, Jilin, China, in 1983, the M.E. degree in computer communication from the Beijing University of Post and Telecommunications, Beijing, China, in 1991, and the Ph.D. degree in software and theory of computer from Jilin University, Changchun, Jilin, China, in 2005. He is currently a Professor and a Ph.D. Supervisor at Jilin University, Director of Key lab under the Ministry of Information Industry, Standing Director of Communication Academy, member of the Computer Science Academy of Jilin Province. His research interests include theory and software technology of network intelligence management, key technology research of wireless mobile communication and services. He participated in 3 projects of NSFC, 863 and funded by National Education Ministry for Doctoral Base Foundation. He has authored 12 projects of NSFC, key projects of Ministry of Information Industry, Middle and Young Science and Technology Developing Funds, Jilin provincial programs, ShenZhen, ZhuHai, and Changchun.

Yuanbo Xu received the B.Sc. and M.S. degrees from the College of Computer Science and Technology, Jilin University, Changchun, China. He is currently working toward the Ph.D. degree at the Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education, Jilin University. His research interests include applications of data mining, recommender system, and mobile computing.



En Wang received the B.E. degree in software engineering in 2011, the M.E. degree in computer science and technology in 2013, and the Ph.D. degree in computer science and technology in 2016 from Jilin University, Changchun, China. He is currently a Lecturer at the Department of Computer Science and Technology, Jilin University. He is a Visiting Scholar at the Department of Computer and Information Sciences, Temple University in Philadelphia, Philadelphia, PA, USA. His current research interests include the efficient utilization of network resources, scheduling and drop strategy in terms of buffer-management, energy-efficient communication between human-carried devices, and mobile crowdsensing.

Jiayu Han received the B.Sc. and M.S. degrees from the College of Computer Science and Technology, Jilin University, Changchun, China. She is currently working toward the Ph.D. degree at the Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education, Jilin University. Her research interests include data mining, data fusion, recommender system, and machine learning.

Zhiwen Yu (SM'11) received the Ph.D. degree in engineering in computer science and technology from the Northwestern Polytechnical University, Xian, China, in 2005. He is currently a Professor at Northwestern Polytechnical University. From February 2007 to January 2009, he was a Research Fellow with the Academic Center for Computing and Media Studies, Kyoto University, Japan. During 2006–2007, he was a Postdoctoral Researcher with the Information Technology Center, Nagoya University, Nagoya, Japan. From November 2009 to October 2010, he was an Alexander von Humboldt Fellow with the Mannheim University, Germany. His research interests include pervasive computing, contextaware systems, human-computer interaction, mobile social networks, and personalization. He is an Associate Editor or Editorial board of IEEE COMMUNICATIONS MAGAZINE, IEEE TRANSACTIONS ON HUMAN MACHINE SYSTEMS, Personal and Ubiquitous Computing, and Entertainment Computing (Elsevier). He was the Guest Editor of *ACM Transactions on Intelligent Systems and Technology*, *ACM Multimedia Systems Journal, Multimedia Tools and Applications* (Springer), Pervasive and Mobile Computing (Elsevier), and Cybernetics and Systems (Taylor&Francis). He is the General Chair IEEE CPSCom15, and IEEE UIC14. He is the Vice Program Chair of PerCom15, the Program Chair of UIC13, and the Workshop Chair of UbiComp11. He has authored and coauthored more than 150 scientific papers in refereed journals and conferences. He is Member of ACM, a Council Member of CCF (China Computer Federation). He received the CPSCom13/GPC12/AMT12/UIC09 Best Paper Award.