A Parameter-free Bayesian-based Framework for Enhancing Top-N Recommender Systems

Rong-qin Xu, Ming-yang Zhou*, Zi-ming Wang, Hao Liao

Shenzhen University, Shenzhen, 518060, China

Abstract

Personalized top-N recommender systems have been investigated widely in decades. In user-item recommendation, an underlying hypothesis exists that users are inclined to consume items that are similar to his/her purchase history. Thus, a core task is to design effective methods to measure user-item preference scores and then to suggest, for each user, a small set of personalized items with high scores. However, little attention was paid to the recommendation of low-score links that are usually unpopular and niche (or brand-new) ones. In this work, based on the Bayesian estimation theory, we propose a novel metric RNR (**R**ecall-to-**N**oise **R**atio) to characterize the ability to recommend both high-score and low-score user-item links. An interesting counterintuitive phenomenon is found that recommending some particular low-score links may achieve higher accuracy than high-score ones. Then we propose a parameter-free framework that leverages an optimal transferring function to redefine the link scores of the state-of-the-art recommendation methods. We analytically prove that the proposed framework could not only optimally improve the recommendation accuracy, but also benefit the recommendation of low-score links, which is verified in empirical experiments. Moreover, the results also show that our framework wins in terms of coverage

*Corresponding author: Ming-yang Zhou (zmy@szu.edu.cn)

metric.
Keywords: Recommender systems, Top-N recommendation, Bayesian estimation, Accuracy, Data mining

1. Introduction

With the explosive growth of online markets, recommender systems have been investigated widely in the academic fields and play an increasingly significant role in E-commerce platforms (e.g., Amazon.com, Netflix) [1, 2, 3]. In order to promote the consumption activities of users on the platforms, one approach is to supply enough products of various brands and qualities to meet the tastes of different users. Consequently, users face the problem of how to choose his/her interested items from the increasing amount of diverse products. Thus, online-business companies devote much energy to design effective recommender systems to help users filter interested items [4, 5]. Besides, recommender systems could also find some potential, yet previously ignored items for users, which further promotes the consumption on the platforms. In some other works, recommender systems also recommend friendship between users in online social networks [6, 7]. Here, we only concern the recommendation in bipartite networks.

In most recommender systems, we usually assign a specific preference score for pairwise user and item, and then suggest a personalized set of N items with the highest scores to each user (top-N recommendation). The key issue is to design effective algorithms to measure the user-item likelihood. Initially, only the topology of user-item bipartite networks is used in recommendation algorithms, including collaborative filtering, matrix factorization and so on [8, 9, 10]. However, these topology-based methods suffer from low coverage and cold-start problems, since most users only consume a few items historically. To solve the problem, other information, for example, personal details of users, item tags and reviews of users on items, is introduced into the recommendation [3, 11, 12]. Yet, the additional information brings about some other problems: How to obtain the hyper-parameters to weigh the importance of different factors? The success of deep learning in the computer vision [13, 14] also inspires scientists to introduce neural networks into recommender systems [12, 3], since deep learning tools (e.g., Tensorflow, Pytorch) provide easy access to tuning the parameters. For example, Twitter uses convolution neural networks to take into account the user information (e.g., age, gender, tags), item attributes (e.g., producer, year, tastes) and the reviews, and the recommendation performs well in real scenarios [15].

However, the aforementioned recommendation methods only recommend items with high scores, neglecting the low-score ones. Low-score items are relevant to the niche products and the particular tastes of users [16]. Previous researches have shown that, in unipartite social networks, long-range links play a critical role on the network dynamics and functions [17, 18], whereas long-range links usually have low similarity scores. Analogically, the low-score links in user-item bipartite networks also play a similar role with that of unipartite social networks, since low-score links help users discover personalized niche products and develop new interests for users [19]. Nevertheless, recommending the niche products for a user only based on the preference score is prohibitive, since irrelevant items also have low scores. Hence, precisely recommending the low-score potential items is still a great challenge in the discipline (See Fig. 1).

Our study takes a different but complementary framework to top-N recommender systems. Unlike most of previous approaches that design different preference score models, we explore the validity of the fundamental underlying hypothesis that whether a user is inclined to consume items with high preference scores to him/her. Inspired by the Bayesian estimation theory and signal-to-noise curves [20, 21], we investigate the characteristic differences of historical consumed and unconsumed items for users, and then propose a metric, named RNR (**R**ecall-to-**N**oise **R**atio), to characterize the ability to recommend potential items. Based on RNR, a novel parameter-free framework is put forwarded to redefine the preference scores of an arbitrary existing method by an one-variable transferring function. We analytically prove the optimum of the transferring function and show that the proposed framework could optimally increase the performance of the existing methods in terms of the recommendation accuracy and item coverage. The main contributions of this work are summarized below:

- We show that a user might consume some particular low-score items, which is previously ignored in the classical recommender systems. Our observation provides a fancy approach to reconsider the design of recommender systems.

- We propose a bayesian-based transferring function to redefine the preference scores of user-item links and theoretically prove that the transferring function could optimally improve the recommendation accuracy of a given method.

- The proposed framework benefits the recommendation of low-score links. More low-score and niche products could be recommended.

- Empirical experiments on real-world datasets verify the effectiveness and generality of RNR framework on different recommender algorithms, such as ItemCF [22], UserCF [23], PureSVD [24] , FISM [25], MultiDAE [26], APR [27] and JCA [28].

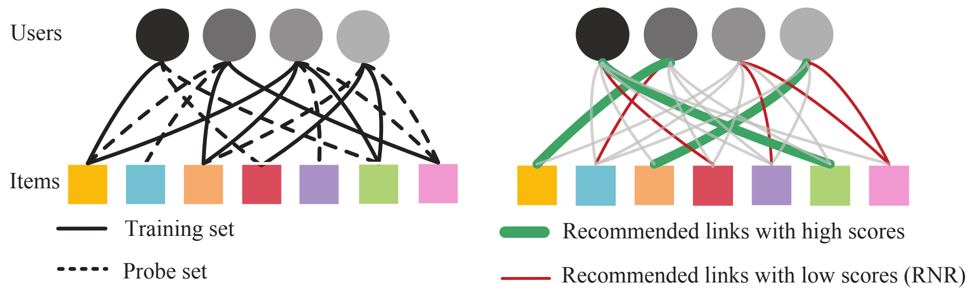
Users
Items
Training set
Probe set
Recommended links with high scores
Recommended links with low scores (RNR)

Figure 1: (a) The original bipartite network is divided into historical training set and future probe set based on the timestamps of links. (b) Schematic of the low-score links recommendation task in a bipartite network. The width of links is proportional to the preference scores.

2. Related works

The problem of recommending future and low-score items touches on the domains of traditional item recommendation and weak link recommendation, which we briefly discuss here.

Item recommendation. In a nutshell, the core task of item recommendation is to accurately recommend a personalized ranking list for each user. Over the past decade, many methods have been developed for top-N recommender systems. Collaborative Filtering (CF) [29, 30] is well-known as one of the most prevalent techniques in recommender systems. The approaches for CF can be roughly categorized into two models: neighborhood models and latent factor models. The neighborhood models make recommendations based on the similarity among users or items such as user-based algorithms [23] and item-based algorithms [22]. Whereas the latent factor models learn hidden patterns from observed ratings using matrix factorization [1, 8]. Apart from CF categories, other methods include random walk based methods, deep learning based methods and so on [2, 5, 31]. Recently, some additional information is introduced into the recommender system, including the intrinsic attributes of users and items and the review information [12, 32]. Besides, to synthesize the additional information, an increasing number of researchers are interested in employing deep learning models (Deep Neural Network, Graph Convolutional Network, Graph embedding) [33, 34] . Whereas most of these methods only facilitate the recommendation of user-item links with high scores, yet fail in the recommendation of low-score ones.

Weak link recommendation. Weak links, also called weak ties in other literature, refer to these links that their endpoints are faraway from either or located in different communities [18, 35]. Here, in bipartite networks without geographical locations nor community structure, weak links are defined as links of low preference scores. Until now, we still lack a strict definition of strong and weak links. In unipartite networks, Watts et al. [36] performed navigation in artificial model networks and found that the capacity of navigation was strongly associated with weak links. Ellison et al. [37] showed that remote friends in Facebook might be more useful than close friends, since most fresh and useful information was transmitted by remote friends. In the search process, Leskovec [38] found that people searched the targets by associating thinking and weak links played an important role. However, most works studied the influence of weak links on the dynamics, the problem of how to recommend weak links lacks insightful investigation. In recommender systems, Wang et al. [39, 35] applied a threshold to distinguish strong (high score) and weak (low score) use-user links, and then utilized supervised methods to optimize the threshold, which greatly enhanced the recommendation accuracy. Apart from the accuracy, some researches focus on improving the novelty and diversity of recommendation algorithms without the loss (or less loss) of accuracy. For example, Zhou et al. [16] put forward a hybrid method to solve diversity-accuracy dilemma of recommendations. Zolaktaf et al. [40]

utilized deep learning method and additional information to investigate the tradeoff of accuracy, novelty, and coverage. However, these works only *implicitly* used the concept of low-score links. The problem of how to *explicitly* identify low-score links requires further investigation.

3. Problem statement

Considering a recommender system, let U and I denote the set of users and items, respectively. Supposing that only the user-item purchase (or viewed, rated) records are prior known that is represented by a bipartite network $G = (U, I, L)$, where each link $(u, i) \in L$ ($u \in U$ and $i \in I$) is attached with a specific purchase timestamp. We denote $n = |U|$ and $m = |I|$ being the size of users and items respectively. A bipartite network is represented by a matrix $R = (r_{ui})_{n\times m}$. If the user u provides feedback for item i, the entry $r_{ui} = 1$; otherwise $r_{ui} = 0$. In order to evaluate the performance of recommendation algorithms, the links of a given network G are divided into training set L^T of size $1 - p^H$ and probe set L^P of size p^H ($0 < p^H < 1$), $L^T \cup L^P = L$ and $L^T \cap L^P = \phi$. $L^T(u)$ and $L^P(u)$ indicate the items that the user u bought in the training set and probe set respectively. The aim is to utilize the links in L^T to recommend the links in L^P as accurately as possible.

Generally, there are mainly three partition methods to divide a dataset, namely, random partition, k-fold cross-validation, time-based partition. For the former two cases, the datasets are randomly divided into training set and probe set. However, it implies that some future links in the training set are used in the recommendation process, which violates the practical scenarios that only historical links could be used to recommend future links. Vidmer et al [41] systematically investigated the influence of time bias on the recommendation. Whereas for the time-based partition, historical links (future links) belong to the training set (probe set), which agrees well with practical scenarios. Thus, in the experiments, we use time-based method to partition the training set and probe set.

After dividing the data into training set and probe set, we should calculate the preference scores only based on the training set. A large number of approaches, including collaborative filtering, matrix factorization, DNN and other methods, are all capable for the calculation of the preference scores (See section 5.2 for more details). When obtaining the preference scores of all nonexisting links, we recommend potential links as those with top-N high scores. Then we use different metrics to evaluate the performance of recommendation, for example, the recall and precision metrics. Here, our concern is how to improve the performance of the current existing methods on recommending low-score links.

4. The proposed framework

In this section, we first present the motivation and the details of RNR framework, and then provide the proof of the optimal RNR transferring function.

4.1. The problem underlying the classical recommendation algorithm

Supposing that s_{ui} is the preference score of a user u towards an item i, obtained by an arbitrary recommendation method (denoted by rec) (See 5.2 for the classical methods). In the state-of-the-art methods, we usually implicitly assume that a higher score link is more likely to be a potential future link, which could be rephrased as

$$p(r_{ui} = 1|s_{ui}) > p(r_{u'i'} = 1|s_{u'i'}), if \quad s_{ui} > s_{u'i'}, \tag{1}$$

where s_{ui} and $s_{u'i'}$ represent the preference scores of two non-existing links in the training set L^T respectively, and $p(r_{ui} = 1|s_{ui})$ is a conditional probability that the user u purchases item i on condition of the preference score being s_{ui}.

According to the bayesian estimation,

$$p(r_{ui} = 1|s_{ui}) = \frac{p(r_{ui} = 1) \cdot p(s_{ui}|r_{ui} = 1)}{p(s_{ui})}, \tag{2}$$

where $p(s_{ui})$ is the probability that a random chosen (existing or non-existing) link in the training set has score s_{ui}. Note that $p(r_{ui} = 1) = |L^T|/(n \times m)$ is a constant representing the probability of an existing link between a random pairwise user-item in the training set. Besides, since real networks are usually sparse and the size of non-existing links is much larger than that of existing links, $p(s_{ui})$ is mainly determined by the score distribution of non-existing links. That is to say, $p(s_{ui}) \approx p_n(s_{ui})$, where $p_n(s_{ui})$ is the probability that a random chosen non-existing link in the training set has score s_{ui}. Thus, we arrive at

$$p(r_{ui} = 1|s_{ui}) = \frac{p(r_{ui} = 1) \cdot p(s_{ui}|r_{ui} = 1)}{p_n(s_{ui})} = c_1 \cdot \frac{p(s_{ui}|r_{ui} = 1)}{p_n(s_{ui})}, \tag{3}$$

where $c_1 = p(r_{ui} = 1)$ is a constant. Neglecting the constant term, we propose **R**ecall-to-**N**oise **R**atio (RNR),

$$RNR(s) = \frac{p(s|r_{ui} = 1)}{p_n(s)} = \frac{p_r(s)}{p_n(s)}, \tag{4}$$

where $p_r(s) = p(s|r_{ui} = 1)$ is the probability that a random chosen existing link in the training set has score s. Note that both $p_r(s)$ and $p_n(s)$ could be obtained only from training set L^T. $RNR(s)$ measures the ability to distinguish real links (i.e., links in the probe set) with the same score.

In the recommender system, we usually assume that the formation of data follows some underlying patterns that rarely change in a short time. Thus,

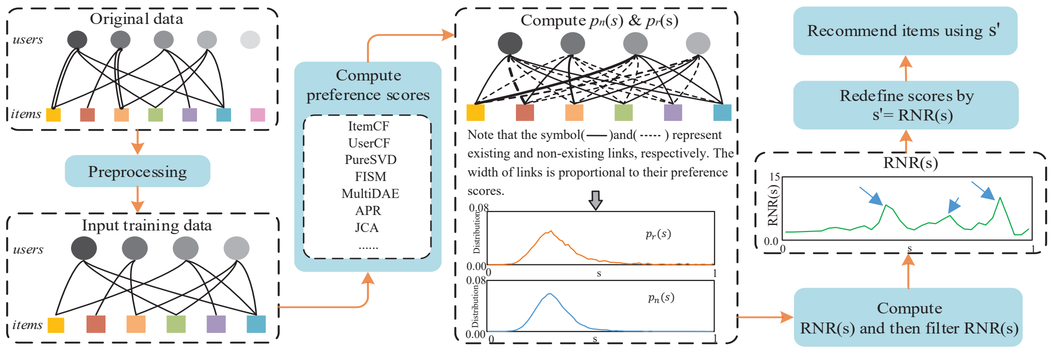
Original data
users
items
Preprocessing
Input training data
users
items
Compute preference scores
ItemCF
UserCF
PureSVD
FISM
MultiDAE
APR
JCA
......
Compute $p_n(s)$ & $p_r(s)$
Note that the symbol(——)and(-----) represent existing and non-existing links, respectively. The width of links is proportional to their preference scores.
$p_r(s)$
$p_n(s)$
Distribution
s
Recommend items using s'
Redefine scores by s'= RNR(s)
RNR(s)
Compute RNR(s) and then filter RNR(s)

Figure 2: The architecture of our proposed RNR framework

we suppose that links in the probe set should have the same (or similar) score distribution with that of the training set. In the following part, we would not differentiate score distributions $p_r(s)$ and $p_n(s)$ for the training and probe sets. The assumption is sound according to the law of large numbers in 180 statistical theory if the time span is small and the size of samples is large enough.

Note that a non-existing link in the training set might be an existing link in the probe set. Given a non-existing link in the training set with the score s_{ui}, the probability that the link is an existing link in the probe set is

$$p' = \frac{|L^P|}{n \times m} \cdot \frac{p_r(s)}{p_n(s)} = \rho \cdot RNR(s), \tag{5}$$

where $\rho = |L^P|/(n \times m)$ is a constant.

Remark 1: Eq. 5 indicates that the recommendation accuracy is determined by $RNR(s)$, rather than the preference score s_{ui} in eq. 1. Therefore, 185 only recommending user-item links of high scores based on the classical assumption (in eq. 1) suffers from some loss of accuracy, motivating us to design better recommender systems.

4.2. The proposed RNR framework

In the section, we use $RNR(s)$ to design a framework to increase recall metric in recommendation. Our framework starts with re-considering the definition of *Recall* that is the ratio of correctly predicted links for the probe set [42].

$$Recall = \frac{l_r}{|L^P|}, \tag{6}$$

where l_r is the size of correctly predicted links. Clearly, higher recall means higher recommendation accuracy, $Recall \in [0, 1]$. Apart from recall, please see section 5.3 for more evaluations.

We propose to divide recall into personalized recall for each user,

$$Recall(u) = \frac{l_r(u)}{|L^P|}, \tag{7}$$

where $l_r(u)$ is the size of correctly predicted links for user u. Actually, the *precision* of user u is defined as $prec(u) = l_r(u)/N = \beta \cdot Recall(u)$, where $\beta = |L^P|/N$. Thus, investigating the personalized recall also means the analysis of precision, and we only discuss the recall issue in this section. Note that $Recall = \sum_{u \in U} Recall(u)$, maximizing the personalized $Recall(u)$ for each user also indicates the maximum of the $Recall$.

Using the formalism of $p_n(s)$ and $p_r(s)$, we can write the size of recommended links as $N = (m - |L^T(u)|) \int_{c(u)}^{s_{max}} p_n(s)ds$, which means nonexisting links (in the training set) with preference score $s_{ij} > c(u)$ are recommended to the user u. $c(u)$ is a personalized parameter for each user that guarantees the size of recommendation list being N, and s_{max} is the maximum score of all links. The size of correctly recommended links could also be wrote as $l_r(u) = |L^P(u)| \cdot \int_{c(u)}^{s_{max}} p_r(s)ds$. Thus, the *Recall(u)* in eq. 7 can be formalized as:

$$\begin{aligned} Recall(u) &= \frac{|L^P(u)| \cdot \int_{c(u)}^{s_{max}} p_r(s)ds}{(m - |L^T(u)|) \cdot \int_{c(u)}^{s_{max}} p_n(s)ds} \\ &= \gamma(u) \cdot \frac{\int_{c(u)}^{s_{max}} p_r(s)ds}{\int_{c(u)}^{s_{max}} p_n(s)ds}, \end{aligned} \tag{8}$$

where $\gamma(u) = \frac{|L^P(u)|}{(m-|L^T(u)|)}$. If $p_r(s) \ll p_n(s)$ at $s > c(u)$, then $Recall(u) \to 0$. Otherwise, $p_r(s) \gg p_n(s)$ gives rise to a high recall.

It is worth noting that eq. 8 and its simplified formulism eq. 7 only recommend items with high preference scores. We generalize eq. 8 by considering links with different scores. Supposing that non-existing links (in the training set) with scores $s \in S_u = (s_1, s_2) \cup (s_3, s_4) ... \cup (s_{2k-1}, s_{2k})$, $s_1 \leq s_2 \leq s_3 \leq ... \leq s_{2k-1} \leq s_{2k}$, are recommended as potential links for user u, following the same analysis of eq. 8, we can calculate the $Recall(u)$ as follows:

$$Recall(u) = \gamma(u) \cdot \frac{\int_{S_u} p_r(s)ds}{\int_{S_u} p_n(s)ds}. \tag{9}$$

200 **Remark 2:** In most previous works, only links with high scores are recommended, which means $S_u = (c(u), s_{max})$, and thus eq. 9 reduces into eq. 8 under the scenario. The generalized $Recall(u)$ in eq. 9 interestingly has the ability to recognize user-item links with low scores.

The upper bound of the $Recall(u)$ in eq. 9 follows as (See eq. 11 for the detailed derivation):

$$Recall(u) \leq \gamma(u) \cdot RNR_{max}(s), \forall s \in S_u. \tag{10}$$

$RNR(s)$ in eq. 10 could be obtained only from the training set. Choosing links with some particular low scores may be better than the high-score ones. 205

Remark 3: The $Recall(u)$ (in eq. 10) could reach the maximum only if we choose links whose preference scores correspond to maximal $RNR(s)$ (See section 4.4 for the proof). That is to say, we choose the optimal set

S_u that satisfies $RNR(s_{ui}) > RNR(s_{vj})$, $\forall s_{ui} \in S_u$ and $\forall s_{vj} \notin S_u$. In the proposed framework, we redefine the score $s'_{ui} = RNR(s_{ui})$ for all links and use the redefined score s'_{ui} to recommend items for users.

Now that the central task is to calculate $RNR(s)$. The calculation of $RNR(s)$ depends only on the score distributions ($p_n(s)$ and $p_r(s)$) of links in training set, and is irrespective of the personalized $c(u)$ in eq. 8 nor other parameters (parameter-free). We can first calculate $p_n(s)$ and $p_r(s)$, and then obtain $RNR(s) = p_r(s)/p_n(s)$. In the computation, we discretize $p_n(s)$ and $p_r(s)$ and calculate them by dividing the scores into b uniform bins with interval length $(s_{max} - s_{min})/b$. In the calculation, some $p_n(s) \to 0$ will introduce much fluctuation into $RNR(s)$ due to the limited size of the networks. In order to reduce the fluctuation, we apply the Gaussian Filter technique to filter $RNR(s)$ curve. The window size of Gaussian Filter is denoted by a parameter w here.

We then propose a generic framework to conduct recommendation based on $RNR(s)$ (See Algorithm 1 and Figure 2):

1. For a given classical recommendation algorithm, we use the algorithm to calculate the preference scores for all user-item links.
2. Compute the discrete $RNR(s)$ and then filter $RNR(s)$ curve by Gaussian Filter.
3. Redefine the score of each user-item link $s'_{ui} = RNR(s_{ui})$ and generate the recommendation list by the re-assigned score s'_{ui}. (The optimal score set S_u is made up of the scores of the recommended links. However, in the recommendation, we do not require to calculate S_u explicitly.)

The highlight of our framework is introducing the $RNR(s)$ to redefine the scores of links. If all high-score links have high $RNR(s)$, our proposed

```
   Input: Implicit feedback matrix $\mathbf{R} \in \mathbf{R}^{n\times m}$ (training set)
   Output: Redefined preference score matrix $\mathbf{M}^{RNR}$
 1 Initialize $b$, $w$ and a given method $rec$;
 2 Compute the preference score matrix $\mathbf{M}^{rec}_{n\times m}$ via $rec$;
 3 Initialize $b$ bins based on the maximum and minimum entries of $\mathbf{M}^{rec}$;
 4 for $i = 1; i \leq b$ do
 5     Count how many existing links ($n_{e,i}$) locate in each bin $i$ and then
       compute $p_r(s_i)$ via $p_r(s_i) = n_{e,i}/|L^T|$;
 6     Count how many non-existing links ($n_{n,i}$) locate in each bin $i$ and
       then compute $p_n(s_i)$ via $p_n(s_i) = n_{n,i}/|O - L^T|$;
 7     if $p_n(s_i) == 0$ then
 8         $RNR(s_i) = 0$;
 9     else
10         Compute $RNR(s_i) = p_r(s_i)/p_n(s_i)$;
11     end
12 end
13 Filter $RNR(s)$ using $GaussianFiltering$ technique;
14 Obtain the redefined matrix $\mathbf{M}^{RNR}$ by $\mathbf{M}^{RNR} = RNR(\mathbf{M}^{rec})$;
15 return $\mathbf{M}^{RNR}$.
```
Algorithm 1: RNR Framework

framework reduces into the original recommendation algorithms. However, if a low-score link has high $RNR(s)$, it could be correctly recognized by our framework.

4.3. Analysis of time complexity

Calculating $RNR(s)$ increases the time consumption in the framework. For the state-of-the-art methods, we need to calculate the preference scores of all non-existing links in which the lower limit of the time complexity is $\Omega(|U| \times |I|)$. In the proposed framework, counting $p_r(s)$ and $p_n(s)$ requires traversing all links between users and items with time complexity $O(|U| \times |I|)$. Thus, introducing $RNR(s)$ in the framework does not increase the time complexity of most classical recommender systems.

4.4. Proof of the optimal RNR(s)

In this subsection, we prove the optimal transferring function $f(s) = RNR(s)$ and provide its relationship with $Recall(u)$. The proof is irrelevant to the design of the experiments. Thus, please skip the proof if you care more about the experiment performance and it would not influence the reading of the paper.

In the proposed framework, we use the redefined score $s'_{ui} = f(s_{ui}) = RNR(s_{ui})$ to recommend items. The scores of the recommended links constitute the optimal score set S_u^f in eq. 10. We argue that any other transferring function $s''_{ui} = f'(s_{ui})$ cannot outperform the performance of $s'_{ui} = f(s_{ui})$. We take two cases to prove the optimal $f(s)$:

Case 1. Supposing that $S_u^{f'} = (s_1^{f'}, s_2^{f'}) \cup (s_3^{f'}, s_4^{f'}) ... \cup (s_{2k-1}^{f'}, s_{2k}^{f'})$ is a score set satisfying $S_u^f \cap S_u^{f'} = \phi$, where S_u^f and $S_u^{f'}$ are the score sets obtained by $f(s) = RNR(s)$ and an arbitrary function $f'(s)$ respectively. Since the length of recommendation list is N, $(m - |L^T(u)|) \cdot \int_{S_u^{f'}} p_n(s)ds = N$, substituting $\int_{S_u^{f'}} p_n(s)ds = N/(m - |L^T(u)|)$ into eq. 9, we have

$$\begin{aligned} Recall(u)^{S_u^{f'}} &= \alpha \int_{S_u^{f'}} p_r(s)ds = \alpha \int_{S_u^{f'}} RNR(s) \cdot p_n(s)ds \\ &< \alpha RNR_{max}(S_u^{f'}) \cdot \int_{S_u^{f'}} p_n(s)ds \\ &= \gamma(u) \cdot RNR_{max}(S_u^{f'}), \end{aligned} \tag{11}$$

where $\alpha = \frac{\gamma(u)(m-|L^T(u)|)}{N}$, and $RNR_{max}(S_u^{f'})$ is the maximum value of $RNR(s), \forall s \in S_u^{f'}$.

Analogously, the $Recall(u)$ of the optimal set S_u^f follows

$$\begin{aligned} Recall(u)^{S_u^f} &= \alpha \int_{S_u^f} RNR(s) \cdot p_n(s)ds \\ &> \alpha RNR_{min}(S_u^f) \cdot \int_{S_u^f} p_n(s)ds \\ &= \gamma(u) \cdot RNR_{min}(S_u^f). \end{aligned} \tag{12}$$

Note that in our proposed method, $RNR(x) > RNR(y), \forall x \in S_u^f, \forall y \in \mathbf{R} - S_u^f$, and $S_u^{f'} \subset \mathbf{R} - S_u^f$. Therefore, $RNR_{min}(S_u^f) > RNR_{max}(S_u^{f'})$ and we arrive at

$$Recall(u)^{S_u^f} > Recall(u)^{S_u^{f'}}. \tag{13}$$

Case 2. Supposing $S_u^f \cap S_u^{f'} = Z \neq \phi$ and $\int_Z p_n(s)ds = C_z$, according to the procedure introduced above, we can also get

$$\begin{aligned} \int_{S_u^f \setminus Z} p_r(s)ds &= \int_{S_u^f \setminus Z} RNR(s) \cdot p_n(s)ds \\ &> RNR_{min}(S_u^f \setminus Z) \cdot \int_{S_u^f \setminus Z} p_n(s)ds \\ &> RNR_{max}(S_u^{f'} \setminus Z) \cdot \int_{S_u^{f'} \setminus Z} p_n(s)ds \\ &> \int_{S_u^{f'} \setminus Z} RNR(s) \cdot p_n(s)ds \\ &= \int_{S_u^{f'} \setminus Z} p_r(s)ds. \end{aligned} \tag{14}$$

Substituting eq. 14 into eq. 9, we get

$$\begin{aligned} Recall(u)^{S_u^f} &= \alpha \cdot \int_{S_u^f \setminus Z} p_r(s)ds + \alpha \cdot \int_Z p_r(s)ds \\ &> \alpha \cdot \int_{S_u^{f'} \setminus Z} p_r(s)ds + \alpha \cdot \int_Z p_r(s)ds \\ &= Recall(u)^{S_u^{f'}}. \end{aligned} \tag{15}$$

Now that we have proved the optimum of the transferring function $f(s) = RNR(s)$ on personalized $Recall(u)$ and that $RNR(s)$ could maximize the $Recall(u)$ for all users. Since $Recall = \sum_{u \in U} Recall(u)$, the $Recall$ is also maximized. Besides, the precision $prec(u) = \beta \cdot Recall(u)$, $\beta = |L^P|/N$, $RNR(s)$ optimizes the precision as well. Therefore, we can conclude that the proposed RNR could utilize the full potential accuracy of most recommendation methods.

5. Experimental results

5.1. Dataset

In order to verify the validity of RNR model, we conduct experiments on four different and representative temporal bipartite networks in which each user-item link has a specific timestamp. The statistics of these four datasets are reported in Table 1 and their descriptions are shown below.

- ML100K[1]: ML100K was collected through the MovieLens website (movielens.umn.edu) during the seven-month period from September 19th, 1997 through April 22nd, 1998. Each user has rated at least 20 movies.
- Delicious[2]: Delicious is derived from the bookmarking and tagging information from a set of 2K users from Delicious social bookmarking

[1]https://grouplens.org/datasets/movielens/

[2]http://socialcomputing.asu.edu/datasets/Delicious

system.

- Lastfm[3]: Lastfm was obtained from the last.fm online music system, where each music artist was listened to by at least 10 users and each user listened to at least 5 artists. The dataset contains music artists listening information from a set of 2K users.

- Wikibooks[4]: This is the bipartite edit network of the French Wikipedia. It contains users and pages from the French Wikipedia, connected by edit events. Each link represents an edit.

Some data may contain multiple links between a user and an item, we only consider a sole link between a pairwise user-item and ignore the weights of links. Since the original datasets are very sparse, the division of training set and probe set may bring about some isolated users and items. In order to simplify the problem, we only keep users or items whose degrees are larger than a threshold. For ML100K dataset, we kept the users of degree≥ 5 and items of degree≥ 1. For Delicious dataset, we kept the users of degree≥ 5 and items of degree≥ 2. For Lastfm dataset, we kept the users of degree≥ 2 and items of degree≥ 2. For Wikibooks dataset, we kept the users of degree≥ 5 and items of degree≥ 1.

5.2. Baselines and parameter setting

There exists a large number of baseline recommendation algorithms. In recent years, deep learning based methods have drawn much attention, including Bayesian Personalized Ranking (BPR) [43], Neural Collaborative Filtering (NCF) [44], Joint Collaborative Autoencoder (JCA) [28], and so on.

[3]http://www.last.fm

[4]http://konect.uni-koblenz.de/networks/edit-frwikibooks

Table 1: Statistic details of four datasets

Dataset	ML100K	Delicious	Lastfm	Wikibooks
# of Users	943	1867	2100	28113
# of Items	1682	69227	18744	2884
# of Links	100000	104799	71064	67613
# of Density	6.30%	0.08%	0.18%	0.08%

However, Maurizio et.al [45] have pointed the controversy of the performance improvement of deep learning based models. To increase the reproductivity of the paper, we compare our method with the state-of-the-art deep learning based methods and the classical collaborative filtering methods. Besides, in the experiment, we do not consider the additional information (e.g., item tags, reviews) in the recommendation for convenience. However, the proposed framework is also capable of other methods.

We select the following seven classical and fresh baseline methods for the studied problem. To reach the best performance, we tuned the key hyper-parameters for each baseline on four datasets and the optimal values of hyper-parameters are reported in the supplementary Table S2.

- ItemCF [22]: **Item**-based **C**ollaborative **F**iltering model first identifies a set of similar items for each of the items that the user has purchased, and then recommends top-N items based on the similarities. We used item-item cosine function to measure the similarity among items.

- UserCF [23]: **User**-based **C**ollaborative **F**iltering model first identifies a set of similar users, and then recommends top-N items based on the purchase history of these users. We used user-user jaccard function to measure the similarity among them.

- PureSVD [24]: **Pure Singular Value Decomposition** method is a matrix factorization model where latent features of both users and items are represented by the principle singular vectors of the user-item matrix. The hyper-parameters, the number of singular values and the number of iterations during SVD, were set to the default values according to the original work [24].

- FISM [25]: **F**actored **I**tem **S**milarity **M**odel is an item-based method and it learns the similarity between items in latent factors, which aims to improve predefined similarity in neighborhood-based methods. We searched the number of latent factors in the range of $[32, 64, 96, 128]$ and the normalization constant was set to 0.50 according to the original work [25].

- MultiDAE [26]: It uses multinomial likelihood to describe the data distribution and re-analyzes the variational autoencoders (VAE) to build better objective function. We tuned the learning rate in the range of $[0.001, 0.005, 0.01, 0.02]$ and the dimensions of latent representation in the range of $[50, 100, 150, 200, 250]$.

- APR [27]: **A**dversarial **P**ersonalized **R**anking introduces adversarial training into the classical Bayesian Personalized Ranking. APR greatly enhances the robustness of the recommendation performance. The embedding size was searched from the range $[16, 32, 64, 96, 128]$ and learning rate was searched from the range $[0.001, 0.005, 0.01, 0.02]$.

- JCA [28]: **J**oint **C**ollaborative **A**utoencoder aims to learn both user-user and item-item correlations simultaneously, which could achieve high robustness for recommender systems. For the regularization trade-off parameter, we searched the best ones over [0.5,0.1,0.05,0.01,0.005, 0.001]. Moreover, the value of hidden layer dimension was tested over $[40, 80, 120, 160, 200, 240]$.

The detailed parameter settings of baseline methods are shown in table S2 in the supplementary. In the proposed framework, the number of uniform bins b and Gaussian Filter window size w is set to 80 and 7 respectively.

5.3. Evaluation metrics

For the top-N recommender systems, we sort the item scores by descending order and recommend the top-N items for each user. The recommendation accuracy is mainly measured by five metrics: Recall, Precison, NDCG (**N**ormalized **D**iscounted **C**umulative **G**ain) [46], MAP (**M**ean **A**verage **P**recision) [47] and ARHR (**A**verage **R**eciprocal **H**it-**R**ank [22]). Apart from accuracy, we use **C**overage and the **A**verage **P**reference **S**core to evaluate the properties of the recommended items. All the results are the average of 10 independent simulations.

- **Recall**. The recall is the ratio of correctly recommended items,

$$Recall@N = \frac{l_r@N}{|L^P|}, \tag{16}$$

  where $l_r@N$ is the correctly recommended items that depends on the length of the recommendation list. For the top-N recommendation, the metric of *Precision* is actually equivalent to the *Recall* (See the analysis of eq. 7). Thus, we only show the *Recall* in the experiment.

- **NDCG**. A limitation of recall metric is that it treats all hits equally regardless of where they appear in the top-N recommendation list. $NDCG$ is the most commonly used list evaluation measure that takes into account the position of the correctly recommended items:

$$DCG(u)@N = \sum_{i=1}^{N} \frac{rel(i)}{log_2(i+1)}, \tag{17}$$

where $rel(i)$ is the relevance level of the item placed at rank position i for each user u, $rel(i) = 1$ for correctly recommended item i and $rel(i) = 0$ for incorrectly recommended item i. Then, $NDCG$ measures the relative DCG of a ranking compared to the best possible ranking: $NDCG(u)@N = DCG(u)@N/IDCG(u)@N$, where $IDCG(u)@N$ is the $DCG(u)@N$ for the ideal ranking for each user u. Finally, the $NDCG$ value for the whole dataset is obtained by $NDCG@N = \frac{\sum_{u \in U} NDCG(u)@N}{|U|}$.

- **MAP**. AP(**A**verage **P**recision) is a ranked precision metric that gives larger credit to correctly recommended items in top rank without any rank-based penalization or discounting. For each user u with recommendation list N, $AP(u)@N$ measures the precision observed in a ranking up to the rank of each relevant item, averaged over the number of relevant items. Thus, $MAP@N$ is computed as,

$$MAP@N = \frac{\sum_{u \in U} AP(u)@N}{|U|}. \tag{18}$$

- **ARHR**. The metric $ARHR@N$ [22] rewards each hit based on where it occurs in the top-N list, which is defined as follows:

$$ARHR@N = \frac{1}{|U|} \cdot \sum_{u \in U} \sum_{i=1}^{hits} \frac{1}{pos_{ui}}, \tag{19}$$

where pos_{ui} is the position of item i in user u's list, $pos_{ui} \in [1, N]$. That is, hits that occur earlier in the recommendation list are weighted higher than those occurring later. Thus, $ARHR@N$ precisely measures the recommendation strength of links in the probe set.

- **Coverage.** In recent years, some researchers have indicated that recommender systems only with high accuracy do not always satisfy users

[48, 49]. Systems with lower coverage may be less valuable for advertisement, since they will be limited in decision making. Denote I_u as the recommended item set for the user u, the coverage [42] is

$$Coverage = |\cup_{u \in U} I_u|/|I|, \tag{20}$$

which characterizes the fraction of items that could be recommended to users.

5.4. Experimental performance

According to the timestamps of links, each dataset is divided into 90% historical links (training set) and 10% future links (probe set). To search the optimal hyper-parameters for each baselines in the training set, we further hold 10% links as a validation set. In Section 5.4.1, we show the empirical $RNR(s)$ to illustrate the effectiveness of the proposed method. In Section 5.4.2, we report the recommendation accuracy. In Section 5.4.3, we analyze the properties of the recommended items. In Section 5.4.4, the parameter sensitivity is presented.

5.4.1. Empirical RNR(s) analysis

We first explore the score distributions $p_r(s)$, $p_n(s)$ and the RNR(s) in Fig. 3. In Fig. 3(a), scores of most links are around 0.32. For the cases of score $s > 0.6$ and $s < 0.1$, $p_n(s) \to 0$, introducing much fluctuation into RNR(s) in Fig. 3(b). Thus, we use the gaussian filter method to reduce the fluctuation. In Fig. 3(b), apart from the high score $s \approx 0.97$, some low scores ($s \approx 0.78 \quad or \quad 0.09$) also have high RNR(s), demonstrating the effectiveness of our proposed scheme and the failure of the classical assumption in eq. 1. We argue that the other baseline methods on other datasets also have the similar performance with Fig. 3(b) (See $RNR(s)$ of other methods on the whole datasets in the supplementary Fig. S1).

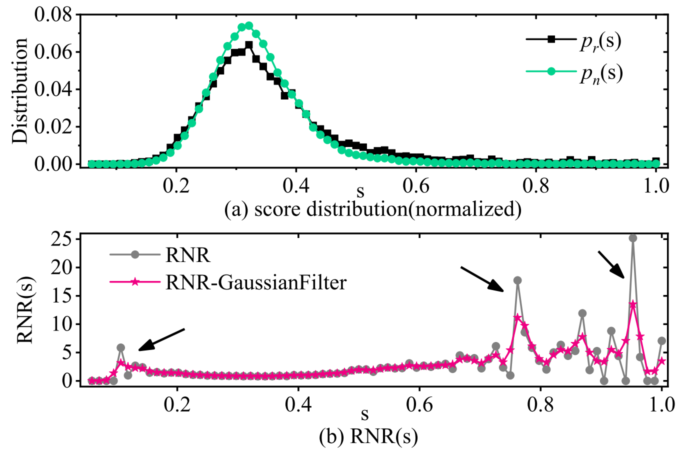

Figure 3: The score distributions of $p_r(s)$ and $p_n(s)$, and the corresponding $RNR(s)$ (including filter-free RNR(s) and filtered RNR(s)) of Delicious dataset, where the user-item scores are calculated by FISM method.

5.4.2. Accuracy performance

Recall performance: Figure 4 reports the recall performance of various methods on four datasets. In Fig. 4, our proposed RNR framework significantly outperforms the original methods in most cases, except the MultiDAE and APR on Delicious dataset. Detailed experiments show that in MultiDAE and APR methods, when we calculate $RNR(s) = p_r(s)/p_n(s)$, $p_n(s) \to 0$ introduces much fluctuation for $RNR(s)$. The fluctuation of $RNR(s)$ largely influences the recall accuracy for MultiDAE and APR methods on Delicious dataset. Whereas for other methods and other datasets, the fluctuation is much smaller, and thus, the recall is improved by our proposed framework. Besides, we also calculate the recall performance for different lengths of recommendation list that has similar observation (See the supplementary Table S2).

NDCG, MAP and ARHR performance: Figure 5 shows the $NDCG@N$ performance as a function of N. Higher $NDCG@N$ is better. In Fig. 5, RNR

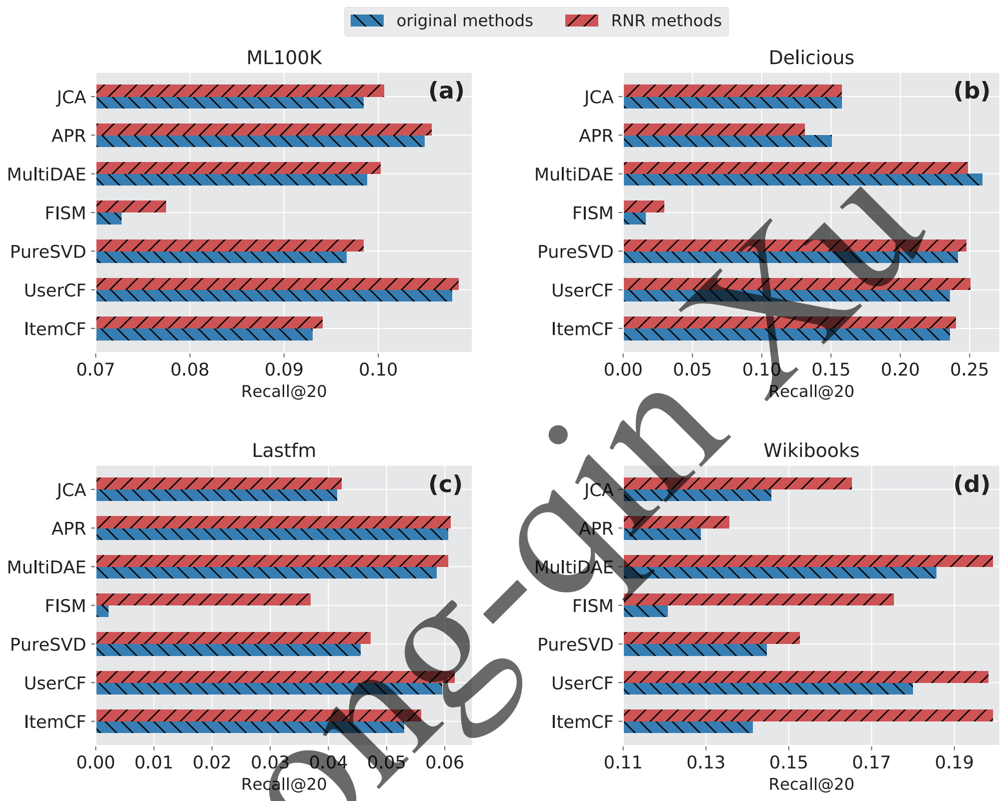

Figure 4: Comparisons of *Recall*@20 results of the RNR recommended links with that of the corresponding original methods on four datasets. (a) ML100K. (b) Delicious. (c) Lastfm. (d) Wikibooks.

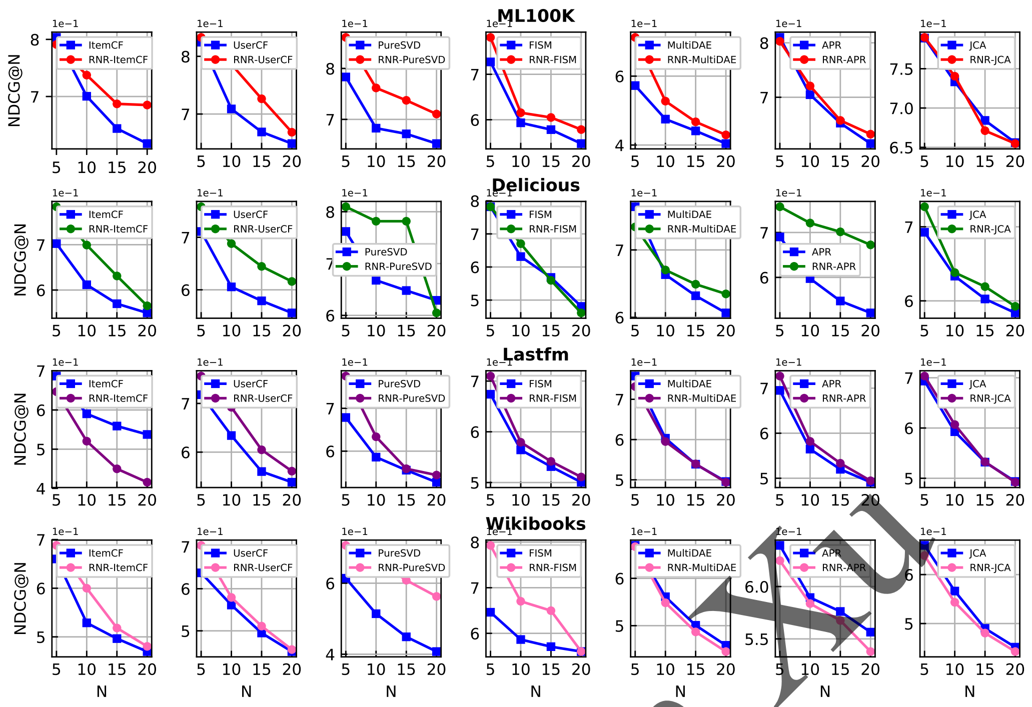

Figure 5: The $NDCG@N$ as a function of N by seven state-of-the-art methods and the corresponding RNR methods on all datasets. Each row represents the results of a dataset.

method outperforms the state-of-the-art methods in most cases. Based on $NDCG@N$, potential items in the probe set occur in the forward position in the top-N recommendation lists on the whole. In practical scenarios, the forward positions usually mean a high click probability and high attraction. Higher $NDCG@N$ implies that our RNR-based method is more valuable in practice. To better reveal the position difference, we show the $MAP@20$ and $ARHR@20$ difference on Lastfm dataset in Fig. 6. Combining Figs. 5 and 6, we can conclude that our method is inclined to push correct items to the forward positions. Apart from the Lastfm, the results hold on in other datasets (Since the results of other datasets are similar to Fig. 6, they are summarized in the supplementary Figs. S3 and S4 for simplifying the paper).

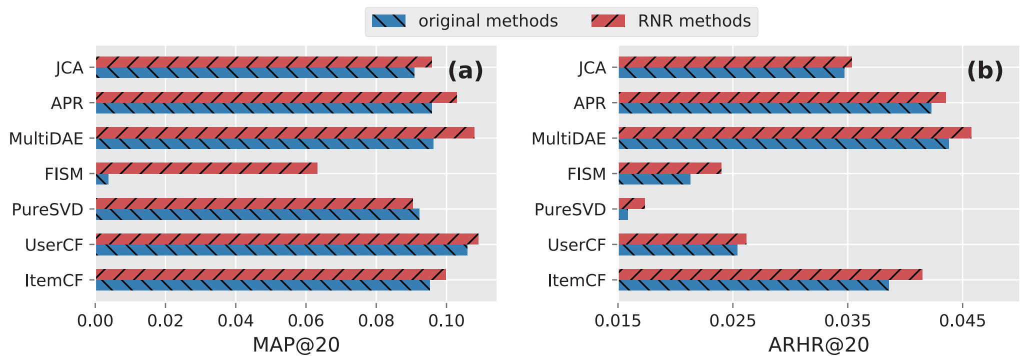

Figure 6: (a) $MAP@20$ results on Lastfm dataset. (b)$ARHR@20$ results on Lastfm dataset.

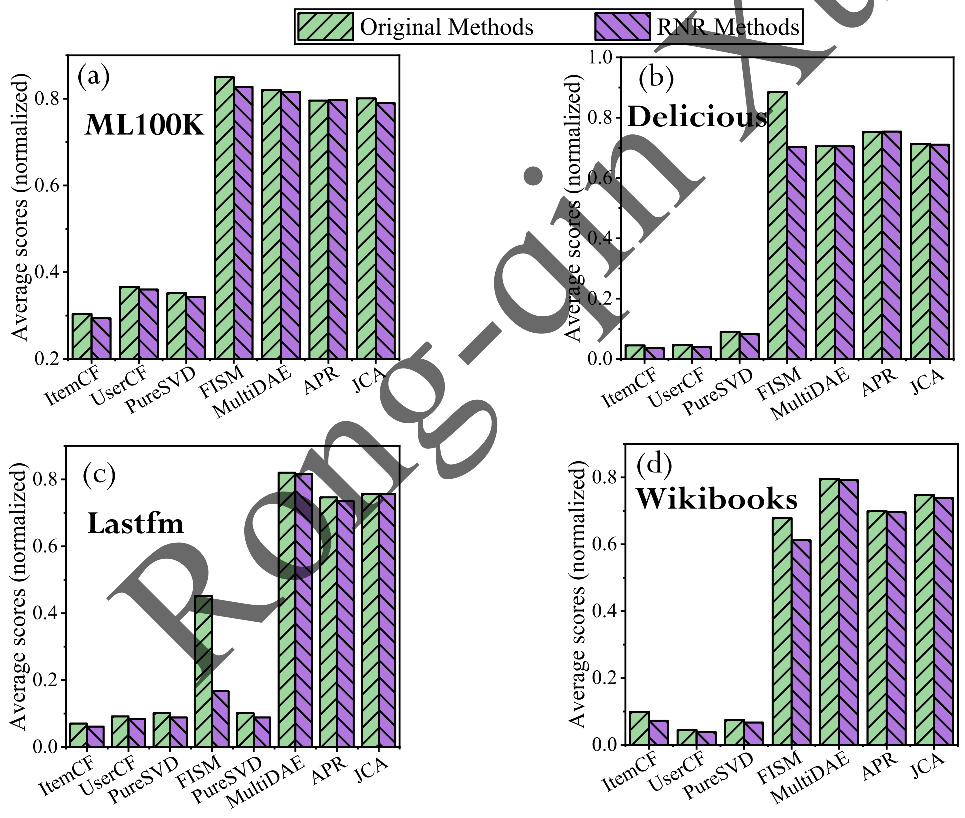

Figure 7: Comparisons of average scores of the RNR recommended links with that of the corresponding original methods on four datasets where $N = 10$. (a) ML100K. (b) Delicious. (c) Lastfm. (d) Wikibooks.

5.4.3. Properties of the recommended items

The correct recommendation of niche and unpopular items is an important issue to surprise users. Here, we treat niche and unpopular items as the links with low scores and analyze the recommendation of low-score links. We investigate the average score of recommended links by our framework, $\bar{s}_{predict} =< s_{ui} >$, where $< ... >$ means the average operator on all recommended items for all users. Smaller $\bar{s}_{predict}$ means more niche items in recommendation list and is better. Figure 7 shows the average scores obtained by RNR methods and the corresponding original methods. Generally, RNR-based method achieves lower average scores than the corresponding original methods on the four datasets, especially for FISM on Delicious and Lastfm datasets. It can be conjectured that more low-score links are detected on the whole, which agrees with Fig. 3.

The coverage metric is strongly associated with the length N of recommendation lists. Larger N means longer recommendation lists for all users and larger coverage. However, in practice scenarios, we usually recommended a limited size of list for users, and hence we set we set $N = 5$ here. Table 2 shows the coverage of all methods, and the proposed method outperforms the classical methods in most cases, implying that more previously ignored low-score items are recommended. Thus, the proposed RNR-based method not only increases the recommendation accuracy, but also benefits the recommendation of unpopular items.

5.4.4. Parameter sensitivity analysis

To study the parameter sensitivity of RNR framework, we have also analyzed the sensitivity of the proposed RNR framework on all datasets. Since there is only one key parameter in the proposed RNR framework, i.e., the number of uniform bins b in the discretization of $RNR(s)$, we discuss the

Table 2: Coverage(%) comparison results on four datasets with the size of recommendation list $N = 5$.

Datasets	ML100K	Delicious	Lastfm	Wikibooks
ItemCF	19.06	74.30	36.78	29.83
RNR-ItemCF	**19.55**	**78.34**	**38.37**	**38.55**
UserCF	16.19	66.54	9.75	11.97
RNR-UserCF	**16.25**	**71.04**	**10.70**	**12.45**
PureSVD	27.98	74.48	19.89	3.36
RNR-PureSVD	**28.36**	**75.20**	**20.40**	**3.59**
FISM	3.60	0.18	0.16	0.92
RNR-FISM	3.60	**0.41**	**1.53**	**0.97**
MultiDAE	58.40	69.57	24.23	23.49
RNR-MultiDAE	**78.44**	**74.51**	**25.33**	**24.71**
APR	**22.30**	**60.39**	**12.98**	**3.36**
RNR-APR	21.32	51.77	12.19	3.30
JCA	**31.15**	**57.27**	**24.47**	**41.31**
RNR-JCA	29.81	54.77	22.40	38.22

influence of b on the experimental performance. As shown in Fig. 8, we present the performance of RNR framework for FISM method on all datasets in terms of $ARHR@20$ evaluation metric. From Fig. 8, $ARHR@20$ increases smoothly with the increase of $b \in (10, 40)$, yet decreases when $b > 40$. The optimal values of b on ML100K, Delicious, Lastfm and Wikibooks dataset are 20, 40, 60, 60, respectively. In the discretization of eq. 4, larger b seems to approach the theoretical $RNR(s)$ better. However, if b is large enough, $p_n(s) \to 0$ in most bins and $RNR(s) \to +\infty$ due to limited size of networks. That is to say, large b would introduce much fluctuation into $RNR(s)$. Thus, we need to choose suitable b for the tradeoff between accuracy and noisy fluctuation. In the empirical study, the ideal b is suggested to be $40 \sim 100$, in which our proposed framework outperforms the state-of-the-art recommendation algorithms in most cases.

6. Conclusion and future work

In this paper, we introduce the Signal to Noise Ratio into recommender systems and propose a parameter-free metric, named RNR(**R**ecall-to-**N**oise-**R**atio), to redefine the preference scores between users and items. We prove that RNR is the optimal transferring function to redefine the user-item preference scores. The proposed RNR has three particular advantages: (1) RNR-based methods have higher recommendation accuracy than the original methods. (2) More items with low scores could be recommended correctly by our method, and simultaneously, the coverage of recommender system is improved. (3) The RNR function could be efficiently calculated only by counting the score difference of existing and non-existing links, which does not improve the time complexity of the classical methods. Moreover, we conduct a comprehensive set of experiments on four real-world datasets and compare

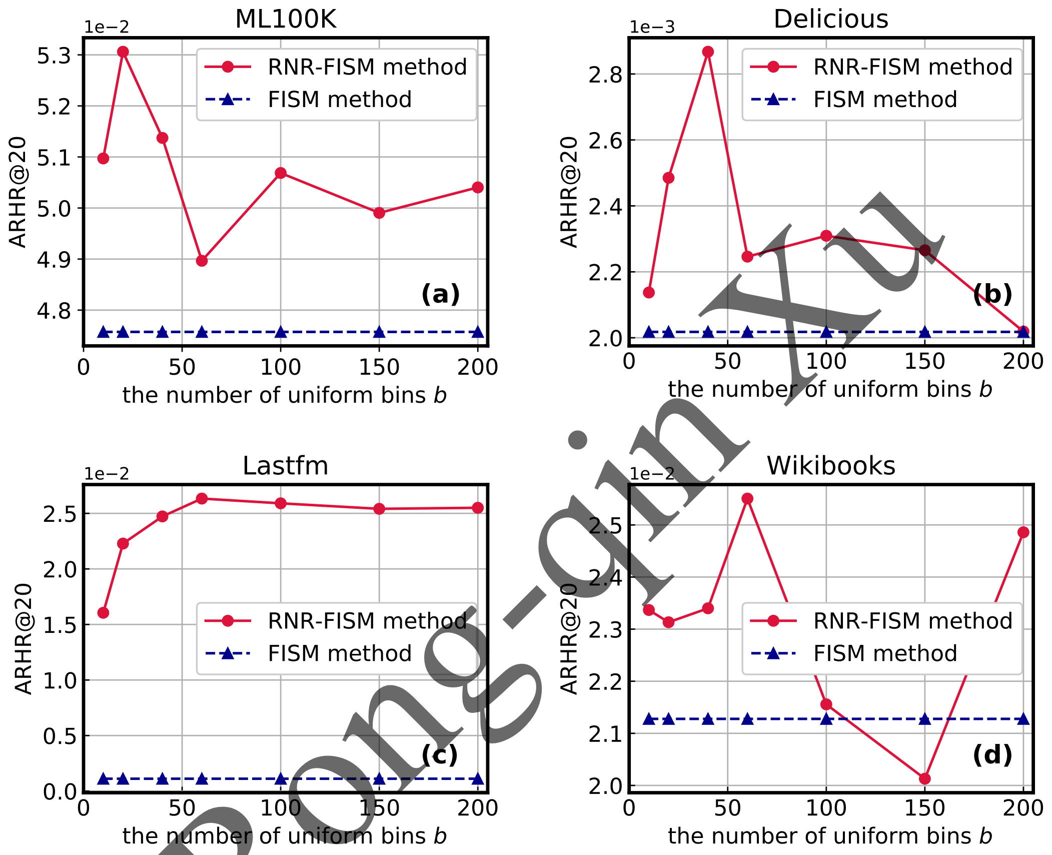

Figure 8: The parameter sensitivity analysis of the number of uniform bins b on four datasets. (a) ML100K. (b) Delicious. (c) Lastfm. (d) Wikibooks.

their performances with the state-of-the-art algorithms. The experiment results validate the efficiency and superiority of our proposed framework.

Furthermore, this work opens up a great deal of opportunities for future research. First, we only consider the case of time-invariant RNR. Whereas in temporal networks, RNR might vary over time, which is a future promising work. Second, calculating RNR requires a large amount of data to reduce the fluctuation. A plausible solution is to utilize deep neural network models to alleviate the problem.

7. acknowledgements

This work was jointly supported by the National Natural Science Foundation of China (Nos. 61703281, 11547040, and 61873171), the Ph.D. Start-Up Fund of Natural Science Foundation of Guangdong Province, China (Nos. 2017A030310374 and 2016A030313036), the Science and Technology Innovation Commission of Shenzhen (No. JCYJ201605 20162743717), Shenzhen Science and Technology Foundation (Nos. JCYJ20150529 164656096 and JCYJ20170302153955969).

Conflicts of Interest: The authors declare no conflicts of interest regarding the publication of this paper.

Authors contribution: M.-Y.Z. and R.-Q. X. designed the research. R.-Q. X. and Zi-M. W. process the data. R.-Q. X. and M.-Y.Z. performed the experiments. M.-Y.Z., R.-Q. X., and H. L. analyzed the data and improved the method. R.-Q. X. wrote the manuscript and all authors have thoroughly proofread and revised the manuscript.