Neurocomputing xxx (xxxx) xxx

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

k-Reciprocal nearest neighbors algorithm for one-class collaborative filtering

Wei Cai[a,b,c], Weike Pan[a,b,c,*], Jixiong Liu[a,b,c], Zixiang Chen[a,b,c], Zhong Ming[a,b,c,*]

[a] *National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University, China*
[b] *Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen University, China*
[c] *College of Computer Science and Software Engineering, Shenzhen University, China*

ARTICLE INFO

Article history:
Received 25 May 2019
Revised 23 October 2019
Accepted 31 October 2019
Available online xxx

Communicated by Dr. Javier Andreu

Keywords:
k-RNN
k-NN
Reciprocal neighbors
One-class collaborative filtering

ABSTRACT

In this paper, we study an important recommendation problem of exploiting users' one-class positive feedback such as "likes", which is usually termed as one-class collaborative filtering (OCCF). For modeling users' preferences beneath the observed one-class positive feedback, the similarity between two users is a key concept of constructing a neighborhood structure with like-minded users. With a well-constructed neighborhood, we can make a prediction of a user's preference to an un-interacted item by aggregating his or her neighboring users' tastes. However, the neighborhood constructed by a typical traditional method is usually asymmetric, meaning that a certain user may belong to the neighborhood of another user but the inverse is not necessarily true. Such an asymmetric structure may result in a less strong neighborhood, which is our major finding in this paper. As a response, we exploit the reciprocal neighborhood among users in order to construct a better neighborhood structure with more high-value users, which is expected to be less influenced by the active users. We then design a corresponding recommendation algorithm called *k*-reciprocal nearest neighbors algorithm (*k*-RNN). Extensive empirical studies on two large and public datasets show that our *k*-RNN performs significantly better than a closely related algorithm with the traditional asymmetric neighborhood and some competitive model-based recommendation methods.

1. Introduction

As an important solution to address the information overload problem, recommendation technologies have attracted a lot of attention in both academia and industry [1]. They can be embedded in many online systems. For example, they can be used to recommend videos in an entertainment website like Netflix and products in an E-commerce website like Amazon. In the community of recommender systems and mobile computing, recommendation with one-class positive feedback [2–4] has recently become an important problem due to the pervasive existence of such behavioral data in various mobile applications such as those for social networking and news feeds services.

For the one-class collaborative filtering (OCCF) problem, some novel algorithms have been proposed in recent years, among which neighborhood-based methods [5] are a series of important approaches. This kind of methods first define a similarity measurement such as Jaccard index and cosine similarity, and then construct a neighborhood with like-minded users according to the similarity values. In particular, *k*-nearest neighbors algorithm (*k*-NN) [5] constructs a neighborhood by taking *k* users or items with the largest similarity values. Importantly, *k*-NN achieves the state-of-the-art performance in many cases despite its simplicity, including OCCF [5], rating prediction [6] and classification tasks [7]. Moreover, it is also appreciated for its good interpretability and easy maintenance, which makes it a commonly used algorithm in real-world recommender systems since the very beginning of collaborative filtering in 1990s [6].

However, *k*-nearest neighborhood is usually asymmetric, which means that a certain user may belong to the neighborhood of another user but the inverse is not necessarily true. This asymmetric phenomenon may cause the fact that some neighbors of a certain user make few contributions to the final recommendation. We call such neighbors as low-value neighbors. Furthermore, for some commonly used similarity measurements such as Jaccard index and cosine similarity, active users or popular items may be

* Corresponding authors at: College of Computer Science and Software Engineering, Shenzhen University, China.
E-mail addresses: caiwei2016@email.szu.edu.cn (W. Cai), panweike@szu.edu.cn (W. Pan), liujixiong@email.szu.edu.cn (J. Liu), chenzixiang2016@email.szu.edu.cn (Z. Chen), mingz@szu.edu.cn (Z. Ming).

https://doi.org/10.1016/j.neucom.2019.10.112
0925-2312/© 2019 Elsevier B.V. All rights reserved.

included in the neighborhood easily. It may not conform to the real situation and thus degrades the recommendation performance.

As a response, we propose to make use of k-reciprocal nearest neighborhood [8] in a recommendation process. Compared with k-nearest neighborhood, k-reciprocal nearest neighborhood serves as a stricter rule to determine whether two users or items are similar, which is symmetric and helpful in screening out the low-value neighbors. One drawback of k-reciprocal nearest neighborhood is the possibility of missing some high-value neighbors due to its strict requirements. In order to cope with this limitation and fully exploit the k-reciprocal nearest neighborhood in the recommendation process, we develop a novel k-reciprocal nearest neighbors algorithm (k-RNN). In k-RNN, we propose to adjust the original similarity in k-reciprocal nearest neighborhood, and then use the ℓ nearest neighborhood with the adjusted similarity for final recommendation, which provides the high-value neighbors an additional opportunity to re-enter the neighborhood.

In order to study the effectiveness of the k-reciprocal nearest neighborhood as it is applied to a recommendation problem for the first time as far as we know, we conduct extensive comparative empirical studies among a closely related algorithm k-NN, some very competitive model-based algorithms, and our k-RNN. Experimental results on two large and real-world datasets show that our k-RNN achieves the best performance, which clearly sheds light on the usefulness of k-reciprocal nearest neighborhood in recommendation algorithms.

We summarize our main contributions as follows: (i) we study an important recommendation problem called one-class collaborative filtering (OCCF); (ii) we identify the asymmetric issue of the neighborhood constructed by a typical traditional neighborhood-based method; (iii) we exploit the reciprocal neighborhood and construct a better neighborhood structure with more high-value users; (iv) we design a novel recommendation algorithm called k-reciprocal nearest neighbors algorithm (k-RNN); and (v) we conduct extensive empirical studies on two large and public datasets to show the effectiveness of our k-RNN.

We organize the rest of the paper as follows. In Section 2, we discuss some closely related works. In Section 3, we first formally define the studied problem, i.e., one-class collaborative filtering, and then present our solution k-reciprocal nearest neighbors algorithm (k-RNN) in detail. In Section 4, we conduct extensive empirical studies with several competitive baseline methods and our k-RNN on two large and public datasets. In Section 5, we conclude the paper with some interesting and promising future directions.

2. Related work

In this section, we discuss some closely related works on one-class collaborative filtering and k-reciprocal nearest neighborhood.

2.1. One-class collaborative filtering

There are mainly two branches of recommendation methods for the studied one-class collaborative filtering problem, including neighborhood-based methods such as k-NN [5], and factorization-based methods such as factored item similarity models (FISM) [3], collaborative denoising auto-encoders (CDAE) [26], restricted Boltzman machine (RBM) [9], Logistic matrix factorization (LogMF) [10], Bayesian personalized ranking (BPR) [11], and matrix completion with preference ranking (PrMC) [12]. Notice that PrMC may not scale well to large datasets in our experiments because the optimization problem of a matrix completion task is usually more complex. As for FISM, CDAE, RBM, LogMF and BPR, the main difference is their objective functions to be optimized and their prediction rules for preference estimation, e.g., the expanded prediction rule in FISM and CDAE, the concise ones in RBM, LogMF and BPR, the energy function in RBM, and the pointwise and pairwise loss functions in FISM, CDAE, LogMF and BPR. Recently, there are also some works based on deep learning such as neural matrix factorization (NeuMF) [13], where the evaluation protocol is different because of the prohibitive time cost of performance evaluation on all the un-interacted items. Though the aforementioned model-based and deep learning based methods have achieved significant success in some complex recommendation tasks with heterogeneous data, they may not be the best for the studied problem of modeling one-class feedback besides their complexity in deployment.

In this paper, we focus on developing novel neighborhood-based methods, which are usually appreciated for their simplicity and effectiveness in terms of development, deployment, interpretability and maintenance.

2.2. *k*-Reciprocal nearest neighborhood

The concepts of reciprocal neighborhood and k-reciprocal neighborhood were first described in [14] and [15]. Later, k-nearest neighborhood was extended to k-reciprocal nearest neighborhood for an important visual application of object retrieval [8]. And some new similarity measurements based on the original primary metric with k-reciprocal nearest neighbors were proposed for an image search task [16]. Recently, an algorithm used to re-rank the initially ranked list with k reciprocal features calculated by encoding k-reciprocal nearest neighbors was developed for person re-identification [17]. There are also some works on designing novel k-NN methods with locality information and advanced distance metrics for classification tasks [18,19]. We can see that existing works on reciprocal neighborhood are mainly for non-personalization tasks, instead of the personalized recommendation task in the studied one-class collaborative filtering problem in this paper.

Although reciprocal neighborhood has been exploited and recognized as an effective technique in computer vision, it is still not clear how it can be applied to the personalized recommendation problem. In this paper, we make a significant extension of the previous works on k-reciprocal nearest neighborhood, and apply it to an important recommendation problem with users' one-class behaviors.

3. *k*-Reciprocal nearest neighbors algorithm

3.1. Problem definition

In our studied problem, we have n users, m items and their associated one-class positive feedback in the form of (user, item) pairs recorded in a set $\mathcal{R}$. Each (user, item) pair $(u, i) \in \mathcal{R}$ means that a user u has expressed a certain positive feedback such as "like" or "thumb up" to an item i. For a typical online system, without loss of generality, we use $\mathcal{U} = \{1, 2, \ldots, n\}$ and $\mathcal{I} = \{1, 2, \ldots, m\}$ to represent the whole set of users and the whole set of items, respectively, and use $\mathcal{I}_u \subseteq \mathcal{I}$ to represent the set of items preferred by user u. Our goal is then to learn users' preferences from these one-class feedback or interactions in $\mathcal{R}$, and generate a personalized ranked list of items from $\mathcal{I} \backslash \mathcal{I}_u$ for each user u as a certain recommendation service. This problem is usually termed one-class collaborative filtering (OCCF), which is different from multi-class collaborative filtering (or collaborative filtering for short) in the famous \$1 Million Netflix Prize. Recently, it has received more attention in the community of recommender systems and mobile computing. We illustrate the studied problem in Fig. 1, and put some commonly used notations and acronyms in the paper in Table 1.

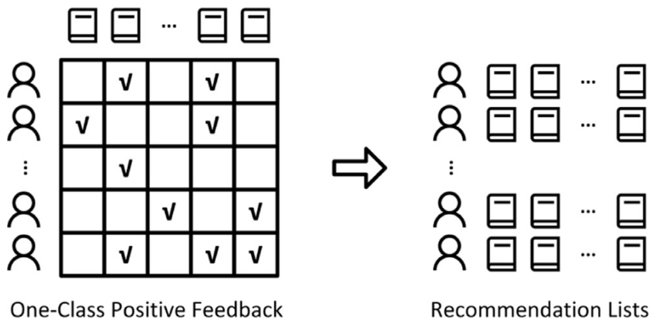


Fig. 1. Illustration of one-class collaborative filtering (OCCF). One-class positive feedback such as users' "likes" to items denoted by "✓" is shown in the left part, and the goal is to generate a personalized recommendation list for each user shown in the right part.

Table 1
Some notations and acronyms and their explanations used in the paper.

Notation	Explanation
n	the number of users
m	the number of items
$u, u', w \in \{1, 2, \ldots, n\}$	user ID
$j \in \{1, 2, \ldots, m\}$	item ID
$\hat{r}_{uj}$	predicted preference of user u to item j
$\mathcal{U}$	the whole set of users
$\mathcal{U}^{te}$	a set of users in test data
$\mathcal{I}$	the whole set of items
$\mathcal{R} = \{(u, i)\}$	a set of one-class feedback in training data
$\mathcal{R}^{va} = \{(u, i)\}$	a set of one-class feedback in validation data
$\mathcal{R}^{te} = \{(u, i)\}$	a set of one-class feedback in test data
$\mathcal{I}_u = \{i\|(u, i) \in \mathcal{R}\}$	a set of items preferred by user u
$\mathcal{I}_u^{te} = \{i\|(u, i) \in \mathcal{R}^{te}\}$	a set of items preferred by user u in test data
s_{uw}	the original similarity between user u and user w
$\tilde{s}_{uw}$	the adjusted similarity between user u and user w
γ	a parameter used in the adjusted similarity
$\mathcal{N}_u^k$	the k-nearest neighborhood of user u
$\mathcal{N}_u^{k\text{-r}}$	the k-reciprocal nearest neighborhood of user u
$\tilde{\mathcal{N}}_u^\ell$	the expanded $\mathcal{N}_u^{k\text{-r}}$
$\ell = \|\tilde{\mathcal{N}}_u^\ell\|$	the size of the expanded neighborhood $\tilde{\mathcal{N}}_u^\ell$
$p(w\|u)$	the position of user w in $\mathcal{N}_u^k$
$\tilde{p}(w\|u)$	the position of user w in $\tilde{\mathcal{N}}_u^\ell$
q	$q = \|\mathcal{R}\|$
OCCF	one-class collaborative filtering
k-NN	k-reciprocal nearest neighbors algorithm
k-RNN	k-nearest neighbors algorithm
PopRank	ranking items via the popularity of the items
FISM	factored item similarity model
RBM	restricted Boltzman machine
PMF	probabilistic matrix factorization
LogMF	Logistic matrix factorization
BPR	Bayesian personalized ranking

3.2. k-Nearest neighborhood

It is well known that the user-based recommendation method is one of the most useful algorithms for the studied one-class collaborative filtering problem. As a key part of the user-based method, the way of constructing a user's neighborhood directly affects the recommendation performance. In this section, we introduce two basic similarity measurements between two users, as well as the selection procedure of the neighbors used in k-nearest neighbors algorithm (k-NN) [5].

3.2.1. Similarity measurement

One way to calculate the similarity between two users is to represent each user as a set of interacted items and measure their similarity via the similarity of the corresponding two sets such as Jaccard index. The Jaccard index between a user u and a user w can then be written as follows,

$$s_{uw} = |\mathcal{I}_u \cap \mathcal{I}_w| / |\mathcal{I}_u \cup \mathcal{I}_w|, \tag{1}$$

where $\mathcal{I}_u$ and $\mathcal{I}_w$ denote the sets of items preferred by the user u and the user w, respectively. Another popular similarity measurement is the well-known cosine similarity, which can be interpreted as a normalized version of Jaccard index, i.e., $|\mathcal{I}_u \cap \mathcal{I}_w| / \sqrt{|\mathcal{I}_u|}\sqrt{|\mathcal{I}_w|}$, for modeling users' one-class feedback. Notice that the performance of Jaccard index and cosine similarity are usually similar in the studied problem. We thus adopt Jaccard index in our empirical studies.

3.2.2. Neighborhood construction

In k-NN, we select the k nearest neighbors of each user in terms of the similarity values. We first denote the position of user w in the neighborhood of user u as follows,

$$p(w|u) = \sum_{u' \in \mathcal{U} \backslash \{u\}} \delta(s_{uu'} > s_{uw}) + 1, \tag{2}$$

where $\delta(x) = 1$ if x is true and $\delta(x) = 0$ otherwise. We can then construct the k-nearest neighborhood of user u as follows,

$$\mathcal{N}_u^k = \{w | p(w|u) \le k, w \in \mathcal{U} \backslash \{u\}\}, \tag{3}$$

which contains k nearest neighbors of user u.

3.3. k-Reciprocal nearest neighborhood

The effectiveness of a recommender system largely depends on the quality of the constructed neighborhood in some algorithms such as the neighborhood-based collaborative filtering methods. Ideally, in k-NN, the k-nearest neighborhood of a user u would include the k most valuable neighbors of user u in terms of closeness and preference prediction accuracy. However, in reality, some low-value neighbors may be included while some high-value neighbors may be left out.

In order to construct the neighborhood more accurately, we propose to make use of k-reciprocal nearest neighborhood, which can be defined as follows,

$$\mathcal{N}_u^{k\text{-r}} = \{w | u \in \mathcal{N}_w^k, w \in \mathcal{N}_u^k\}, \tag{4}$$

where the size of the k-reciprocal nearest neighborhood of user u may be smaller than k, i.e., $|\mathcal{N}_u^{k\text{-r}}| \le k$. We can see that the k-reciprocal nearest neighborhood $\mathcal{N}_u^{k\text{-r}}$ in Eq. (4) is defined by the k-nearest neighborhood $\mathcal{N}_u^k$ shown in Eq. (3).

There are two differences between the k-nearest neighborhood $\mathcal{N}_u^k$ and the k-reciprocal nearest neighborhood $\mathcal{N}_u^{k\text{-r}}$. One is that the latter makes a stricter requirement on whether two users are neighbors, so fewer low-value neighbors would be included in $\mathcal{N}_u^{k\text{-r}}$. The other is that the users in $\mathcal{N}_u^{k\text{-r}}$ depend on both user u and user w. We illustrate a real example of $\mathcal{N}_u^k$ and $\mathcal{N}_u^{k\text{-r}}$ from a real-world dataset MovieLens 20M [20] in Fig. 2. Obviously, the users in $\mathcal{N}_u^{k\text{-r}}$ have higher values than the other users in $\mathcal{N}_u^k$.

Due to the characteristics of some similarity measurements such as Jaccard index and cosine similarity, active users are more likely to be included in $\mathcal{N}_u^k$ than the others. This phenomenon doesn't accurately reflect real situations, which may thus degrade the accuracy of some recommendation algorithms. But in the k-reciprocal nearest neighborhood, for an active user u, $\mathcal{N}_u^{k\text{-r}}$ requires both conditions $u \in \mathcal{N}_w^k$ and $w \in \mathcal{N}_u^k$ instead of the condition $w \in \mathcal{N}_u^k$ only, which is less likely influenced by a user's activeness and can then help construct a more reliable and accurate neighborhood. In Fig. 3, we illustrate the relationship between users' activeness and the average number of times that users with different activeness appear in k-nearest neighborhood and k-reciprocal nearest neighborhood of other users on the first copy of MovieLens 20M. Apparently, users' activeness has less influence on the k-reciprocal nearest neighborhood than on the k-nearest neighborhood.

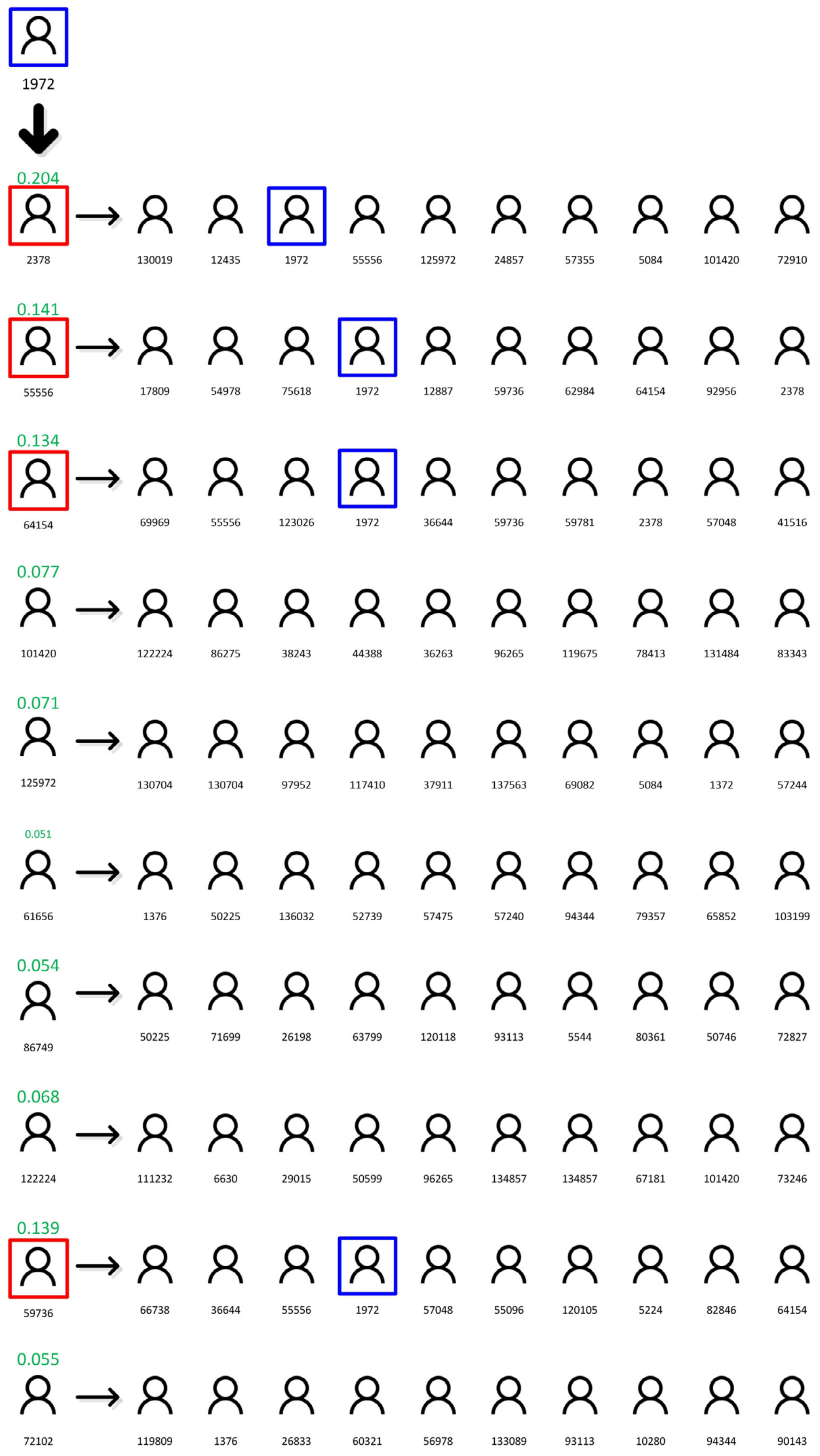


Fig. 2. Illustration of k-nearest neighborhood and k-reciprocal nearest neighborhood. For the user marked with a blue box on the top left corner, we have ten nearest neighbors shown in the left column, where four users marked with red boxes are the reciprocal neighbors. Each number above each of those ten neighbors represents the value of that neighbor, e.g., $|(\mathcal{I}_{2378} \backslash \mathcal{I}_{1972}) \cap \mathcal{I}^{te}_{1972}| / |(\mathcal{I}_{2378} \backslash \mathcal{I}_{1972}) \cup \mathcal{I}^{te}_{1972}|$ for user 1972 and user 2378. Notice that we also include ten nearest neighbors of each of the ten neighboring users on the right.

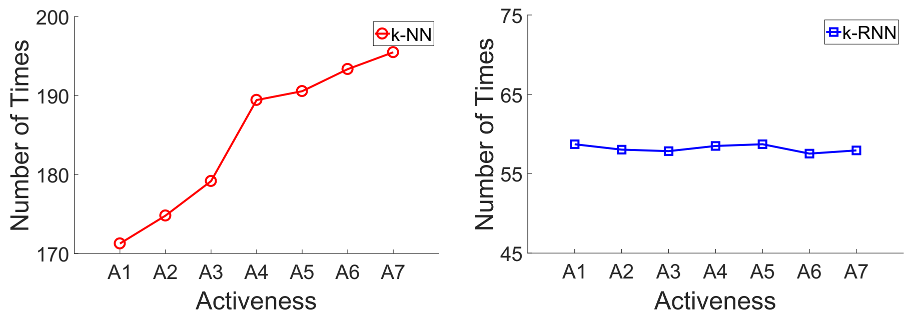


Fig. 3. The relationship between users' activeness denoted by $|\mathcal{I}_u|$ and the average number of times the users with the corresponding activeness appear in k-nearest neighborhood (left) $\mathcal{N}_u^k$ and k-reciprocal nearest neighborhood (right) $\mathcal{N}_u^{k\text{-r}}$ of other users on the first copy of MovieLens 20M. Notice that A1=[30,39), A2=[40,49), A3=[50,59), A4=[60,69), A5=[70,79), A6=[80,89), A7=[90,99), and we fix $k = 200$.

3.4. The recommendation algorithm

The usage of k-reciprocal nearest neighborhood $\mathcal{N}_u^{k\text{-r}}$ is diverse. In this section, we describe one way to use $\mathcal{N}_u^{k\text{-r}}$ in a user-based recommendation method in detail.

3.4.1. Similarity adjustment and neighborhood expansion

The k-reciprocal nearest neighborhood is able to screen out low-value neighbors with a strict rule, but a small number of high-value neighbors may also be omitted meanwhile. In addition, the size of $\mathcal{N}_u^{k\text{-r}}$ is uncertain and may be small in some cases. In order to address these two issues, we propose a novel and appropriate way to make use of $\mathcal{N}_u^{k\text{-r}}$.

Firstly, we adjust the original similarity as follows,

$$\tilde{s}_{uw} = \begin{cases} (1+\gamma)s_{uw}, & u \in \mathcal{N}_w^{k\text{-r}} \\ s_{uw}, & u \notin \mathcal{N}_w^{k\text{-r}} \end{cases}, \tag{5}$$

where $\tilde{s}_{uw}$ is the adjusted similarity between user u and user w. Notice that $\gamma \geq 0$ is a parameter which determines the magnitude of the adjustment. In particular, when $\gamma > 0$ and $u \in \mathcal{N}_w^{k\text{-r}}$, the similarity is amplified in order to emphasize its importance; and when $\gamma = 0$, the adjusted similarity is reduced to the original one.

Similarly, we have the position of user w in the context of user u with the new similarity as follows,

$$\tilde{p}(w|u) = \sum_{u' \in \mathcal{U}\backslash\{u\}} \delta(\tilde{s}_{uu'} > \tilde{s}_{uw}) + 1. \tag{6}$$

With the new position $\tilde{p}(w|u)$, we can construct an expanded neighborhood,

$$\tilde{\mathcal{N}}_u^{\ell} = \{w|\tilde{p}(w|u) \leq \ell, w \in \mathcal{U}\backslash\{u\}\}, \tag{7}$$

where $|\tilde{\mathcal{N}}_u^{\ell}| = \ell$. Notice that when s_{uw} or ℓ is large enough, we may have $w \in \tilde{\mathcal{N}}_u^{\ell}$ even though $w \notin \mathcal{N}_u^{k\text{-r}}$. This is actually very important, because it provides an additional opportunity to the screened-out but high-value users to re-enter the neighborhood again.

Notice that the parameter γ in Eq. (5) adjusts the ratio of $\mathcal{N}_u^{k\text{-r}}$ to $\tilde{\mathcal{N}}_u^{\ell}$ when constructing the final neighborhood, and it also controls the influence of the users in the neighborhood on the prediction results. In our empirical studies, we fix $\gamma = 1$ for simplicity.

3.4.2. Prediction rule

We hope that the users in the k-reciprocal nearest neighborhood $\mathcal{N}_u^{k\text{-r}}$ will have higher impacts on the prediction process. Hence, we directly use the adjusted similarity $\tilde{s}_{uw}$ to predict the preferences of user u to the un-interacted items instead of using the original similarity s_{uw}. Mathematically, we have the new prediction rule as follows,

$$\hat{r}_{uj} = \sum_{w \in \tilde{\mathcal{N}}_u^{\ell} \cap \mathcal{U}_j} \tilde{s}_{uw}. \tag{8}$$

We illustrate a running example of a user (ID: 18753) from MovieLens 20M in Fig. 4. In Fig. 4, the proposed recommendation algorithm, i.e., k-RNN, can recommend five preferred items with the final neighborhood $\tilde{\mathcal{N}}_u^{\ell}$ and the adjusted similarity $\tilde{s}_{uw}$. Notice that the algorithm with the original similarity s_{uw} and the neighborhood $\mathcal{N}_u^k$ generates five items that the user does not like, which is shown on top of Fig. 4.

3.4.3. The algorithm

We describe the detailed steps of the k-reciprocal nearest neighbors algorithm (k-RNN) in Algorithm 1. Firstly, we calculate the similarity and construct the k-nearest neighborhood of each user via Eqs. (1)–(3), which correspond to line 3 to line 9 in the algorithm. Secondly, we estimate the adjusted similarity $\tilde{s}_{uw}$ and construct an expanded reciprocal nearest neighborhood $\tilde{\mathrm{N}}_u^{\ell}$ via Eqs. (5) and (7), which are shown from line 10 to line 21 in the

Algorithm 1 The k-reciprocal nearest neighbors algorithm (k-RNN).

```
1: Input: $\mathcal{R} = \{(u,i)\}$
2: Output: A personalized ranked list of items for each user
3: // Construct the $k$-nearest neighborhood
4: for $u = 1$ to $n$ do
5:     for $w = 1$ to $n$ do
6:         $s_{uw} = |\mathcal{I}_u \cap \mathcal{I}_w|/|\mathcal{I}_u \cup \mathcal{I}_w|$
7:     end for
8:     Take $k$ users with largest $s_{uw}$ as $\mathcal{N}_u^k$ for user $u$
9: end for
10: // Construct the expanded $k$-reciprocal nearest neighborhood
11: for $u = 1$ to $n$ do
12:     for $w = 1$ to $n$ do
13:         $s_{uw} = |\mathcal{I}_u \cap \mathcal{I}_w|/|\mathcal{I}_u \cup \mathcal{I}_w|$
14:         if $u \in \mathcal{N}_w^k$ and $w \in \mathcal{N}_u^k$ then
15:             $\tilde{s}_{uw} = (1+\gamma)s_{uw}$   // $w \in \mathcal{N}_u^{k\text{-r}}$
16:         else
17:             $\tilde{s}_{uw} = s_{uw}$   // $w \notin \mathcal{N}_u^{k\text{-r}}$
18:         end if
19:     end for
20:     Take $\ell$ users with largest $\tilde{s}_{uw}$ as $\tilde{\mathcal{N}}_u^{\ell}$ for user $u$
21: end for
22: // Prediction of each user $u$'s preference to each item $j$
23: for $u = 1$ to $n$ do
24:     for $j \in \mathcal{I}\backslash\mathcal{I}_u$ do
25:         $\hat{r}_{uj} = \sum_{w \in \mathcal{U}_j \cap \tilde{\mathcal{N}}_u^{\ell}} \tilde{s}_{uw}$
26:     end for
27:     Take $N$ items with largest $\hat{r}_{uj}$ as the recommendation list for user $u$
28: end for
```

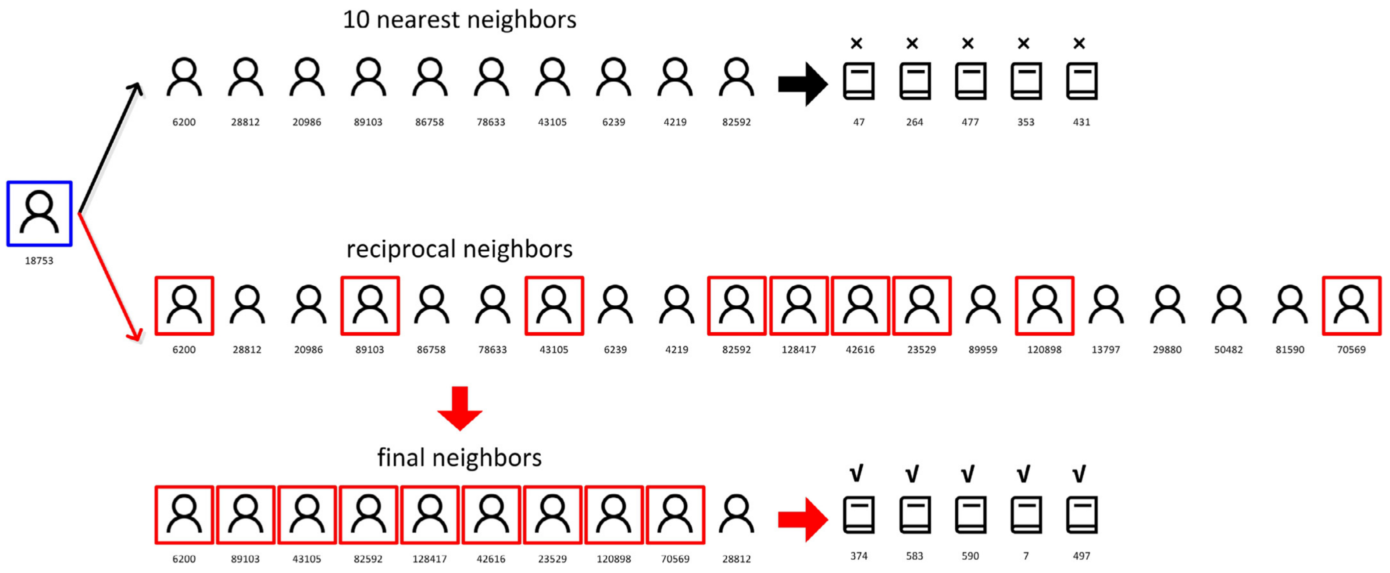


Fig. 4. A running example of two recommendation lists via ten-nearest neighbors (top) and ten-reciprocal nearest neighbors (bottom), respectively. Notice that the users marked with red boxes are the reciprocal neighbors of the user marked in a blue box.

algorithm. Finally, we make predictions of users' preferences to the items via the prediction rule in Eq. (8) shown in the last block from lines 22 to line 28.

For top-N recommendation, by checking whether items in $\mathcal{I}_u$ are in $\mathcal{I}_w$ one by one, the time complexity of calculating $|\mathcal{I}_u \cap \mathcal{I}_w|$ is $\Theta(|\mathcal{I}_u|)$ if we use hash set to store $\mathcal{I}_u$ and $\mathcal{I}_w$. And $|\mathcal{I}_u \cup \mathcal{I}_w|$ can be calculated by $|\mathcal{I}_u \cup \mathcal{I}_w| = |\mathcal{I}_u| + |\mathcal{I}_w| - |\mathcal{I}_u \cap \mathcal{I}_w|$ in $\Theta(1)$. So the calculation of s_{uw} in line 6 and line 13 can be done in $\Theta(|\mathcal{I}_u|)$. Similarly, the time complexity of line 25 is $\Theta(|\mathcal{U}_j|)$. Line 8 can be calculated in $\Theta(n\log k)$ if we use heap for optimization. For convenience, we denote $q = |\mathcal{R}| = \sum_{u\in\mathcal{U}} |\mathcal{I}_u| = \sum_{j\in\mathcal{I}} |\mathcal{U}_j|$. Thus, the time complexity of line 3 to line 9 is $\sum_{u\in\mathcal{U}}(\sum_{w\in\mathcal{U}} \Theta(|I_u|) + \Theta(n\log k)) = \Theta(nq + n^2\log k)$. If we use hash set to store $\mathcal{N}_u$, we can judge whether $u \in \mathcal{N}_w^k$ and $w \in \mathcal{N}_u^k$ is true in $\Theta(1)$, so line 10 to line 21 would cost $\sum_{u\in\mathcal{U}}(\sum_{w\in\mathcal{U}} \Theta(|I_u|) + \Theta(n\log \ell)) = \Theta(nq + n^2\log \ell)$. Taking a few (e.g., $N = 5$ in the experiments) items with the largest $\hat{r}_{uj}$ would cost $\Theta(m\log N)$ in line 27, so the time complexity of lines 22 to 28 is $\sum_{u\in\mathcal{U}}(\sum_{j\in\mathcal{I}} \Theta(|\mathcal{U}_j|) + \Theta(m\log N)) = \Theta(nq + nm\log N)$. Hence, the overall time complexity of the algorithm is $\Theta(nq + n^2\log k + n^2\log \ell + nm\log N)$, where k is usually a scalar multiple of ℓ, i.e., $\ell = \Theta(k)$. Thus, we have $\Theta(nq + n^2\log k + n^2\log \ell + nm\log N) = \Theta(nq + n^2\log k + nm\log N)$, which is equal to the time complexity of recommendation with k-nearest neighborhood. In real-world applications, we usually have $nq \gg n^2\log \ell$, so the additional $n^2\log \ell$ has little impact on the final time cost. In addition, because we usually have $nq \gg n^2\log \ell$, $nq \gg n^2\log k$ and $nq \gg nm\log N$, it is mainly nq that affects the running time. When n increases, n and q in $\Theta(nq)$ increase simultaneously, so the increase of the running time is super linear. Similarly, as m increases, q increases and so does the running time. However, the increase of ℓ, k and N has little effect on the running time. Hence, k-RNN is very similar to k-NN in scalability. Notice that we have to store $\mathcal{N}_u$ and $\tilde{\mathcal{N}}_u$ in this algorithm, so the memory complexity is $\Theta(nk + n\ell) = \Theta(nk)$, which is also equal to the memory complexity of only storing $\mathcal{N}_u$.

4. Experiments

4.1. Datasets

In our empirical studies, we use two large and public real-world datasets, i.e., MovieLens 20M (denoted as ML20M) and

Table 2
Statistics of the first copy of each dataset used in the experiments. Notice that n is the number of users and m is the number of items, and $|\mathcal{R}|$, $|\mathcal{R}^{va}|$ and $|\mathcal{R}^{te}|$ denote the numbers of (user, item) pairs in training data, validation data and test data, respectively.

Dataset	n	m	$\|\mathcal{R}\|$	$\|\mathcal{R}^{va}\|$	$\|\mathcal{R}^{te}\|$
ML20M	138,493	27,278	5,997,245	1,999,288	1,998,877
Netflix	50,000	17,770	3,551,369	1,183,805	1,183,466

Netflix. Both ML20M and Netflix contain records in the form of (user, item, rating) triples. Each (user, item, rating) triple (u, i, r_{ui}) means that a user u assigns a rating r_{ui} to an item i, where $r_{ui} \in \{0.5, 1, 1.5, \ldots, 5\}$ in ML20M and $r_{ui} \in \{1, 2, 3, 4, 5\}$ in Netflix. ML20M contains about 20 million records with 138,493 users and 27,278 items, and Netflix contains about 100 million records with 480,189 users and 17,770 items. We randomly sample 50,000 users in Netflix, and keep all the rating records associated with these users. In order to simulate one-class feedback, we follow [21] and keep all the records (u, i, r_{ui}) with $r_{ui} \geq 4$ as one-class feedback.

For ML20M and Netflix, we randomly pick about 60% (user, item) pairs as training data, about 20% (user, item) pairs as test data and the remaining about 20% (user, item) pairs as validation data. We repeat this process for three times, and obtain three copies of data. The statistics of the processed datasets are shown in Table 2. In our experiments, we use the averaged performance of the recommendation algorithms on these three copies of datasets as the final performance. The data[1] and code[2] used in the experiments are publicly available.

4.2. Evaluation metrics

In order to evaluate the performance of the collaborative filtering algorithms, we use some commonly adopted ranking-oriented evaluation metrics for top-N recommendations [22], i.e., precision, recall, F1, normalized discounted cumulative gain (NDCG) and 1-call.

We denote the personalized ranked list of items for each user u as $\mathcal{I}_u^N = \{i_1, i_2, \ldots, i_N\}$, where $\mathcal{I}_u^N \subseteq \mathcal{I}\backslash\mathcal{I}_u$.

[1] http://csse.szu.edu.cn/staff/panwk/publications/kRNN/.
[2] https://github.com/mdyx/k-RNN/.

- Prec@N: The percision of user u is defined as $Prec_u@N = \frac{|\mathcal{I}_u^N \cap \mathcal{I}_u^{te}|}{N}$, where $|\mathcal{I}_u^N \cap \mathcal{I}_u^{te}|$ denotes the number of items we recommend to user u that appear in the test data. Then, Prec@N can be defined as $Prec@N = \frac{1}{|\mathcal{U}^{te}|}\sum_{u\in\mathcal{U}^{te}} Prec_u@N$.
- Rec@N: The recall of user u is defined as $Rec_u@N = \frac{|\mathcal{I}_u^N \cap \mathcal{I}_u^{te}|}{|\mathcal{I}_u^{te}|}$. Then, Rec@$N$ can be defined as $Rec@N = \frac{1}{|\mathcal{U}^{te}|}\sum_{u\in\mathcal{U}^{te}} Rec_u@N$.
- F1@N: The F1 of user u is defined as $F1_u@N = 2 \times \frac{Prec@N \times Rec@N}{Prec@N + Rec@N}$, which combines $Prec@N$ and $Rec@N$ in one single equation. Then, F1@N can be defined as $F1@N = \frac{1}{|\mathcal{U}^{te}|}\sum_{u\in\mathcal{U}^{te}} F1_u@N$.
- NDCG@N: The NDCG of user u is defined as $NDCG_u@N = \frac{DCG_u@N}{IDCG_u}$, where $DCG_u@N = \sum_{j=1}^{N} \frac{2^{\delta(i_j \in \mathcal{I}_u^{te})}-1}{\log(j+1)}$ and $IDCG_u = \sum_{j=1}^{\min\{N,|\mathcal{I}_u^{te}|\}} \frac{1}{\log(j+1)}$. Then, NDCG@$N$ can be defined as $NDCG@N = \frac{1}{|\mathcal{U}^{te}|}\sum_{u\in\mathcal{U}^{te}} NDCG_u@N$.
- 1-call@N: $1\text{-}call_u@N = \delta(\mathcal{I}_u^N \cap \mathcal{I}_u^{te} \neq \emptyset)$, where $\mathcal{I}_u^N \cap \mathcal{I}_u^{te} \neq \emptyset$ means that at least one item in the test data is recommended to user u. Then, 1-call@N can be defined as $1\text{-}call@N = \frac{1}{|\mathcal{U}^{te}|}\sum_{u\in\mathcal{U}^{te}} 1\text{-}call_u@N$.

In particular, we set $N = 5$ because most users may not check all the recommended items except the few top ones on the list. Hence, we have Pre@5, Rec@5, F1@5, NDCG@5 and 1-call@5 for performance evaluation.

4.3. Baselines and parameter configurations

In the empirical studies, in order to study the effectiveness of our k-RNN, we include the following commonly used and competitive baseline methods:

- ranking items via the popularity (PopRank) is a commonly used simple and basic recommendation method, which is solely based on the popularity of the items and is thus a non-personalized method;
- k-nearest neighbors algorithm (k-NN) [5] is a closely related recommendation method with asymmetric neighborhood for one-class collaborative filtering, which is used for direct comparative study of the effectiveness of the symmetric neighborhood in our k-RNN;
- factored item similarity model (FISM) [3] learns the similarities among items based on the observed one-class interactions between users and items;
- collaborative denoising auto-encoders (CDAE) [26] is a well-known recommendation method for one-class collaborative filtering, which combines a user-specific latent feature vector for personalization and reconstructs the user's interaction vector via auto-encoders;
- restricted Boltzman machine (RBM) [9] has recently been demonstrated its capability of modeling one-class feedback;
- probabilistic matrix factorization (PMF) [23] is a well-studied model-based method for 5-star numerical ratings with a square loss, which is adapted to our studied problem by sampling some negative feedback [2];
- Logistic matrix factorization (LogMF) [10] is a recent method for modeling one-class feedback; and
- Bayesian personalized ranking (BPR) [11] is a very competitive model-based recommendation method with pairwise preference assumption.

For k-NN, we use Jaccard index as the similarity measurement between two users or two items in k-NN(user) and k-NN(item), respectively, and take $k = 100$ most similar users or items as the neighborhood of user u or item i for preference prediction of user u to item j, where $j \in \mathcal{I}\backslash\mathcal{I}_u$. For FISM, RBM, PMF and LogMF, we follow [3] and randomly sample $3|\mathcal{R}|$ un-interacted (user, item) pairs as negative feedback. For FISM, PMF, LogMF and BPR, we fix the number of latent dimensions $d = 100$ (the same with that of k in k-NN), the learning rate $\gamma = 0.01$, and search the best tradeoff parameter $\lambda \in \{0.001, 0.01, 0.1\}$ and the iteration number $T \in \{10, 20, 30, \ldots, 1000\}$ by checking the NDCG@5 performance on the first copy of the validation data. For RBM, we mainly follow [24,25]. Specifically, we run Gibbs sampling in one step, and fix the number of hidden units 100, the batch size 16, the learning rate 0.05, the momentum 0.9, the number of epochs 1000 with early stop strategy, and choose the best regularization parameter of weight decay from {0.0005, 0.001, 0.005, 0.01} via the performance of NDCG@5 on the first copy of the validation data. For CDAE, we mainly follow the guidance in the original paper [26]. Specifically, we set the hidden dimension 100 for fair comparison with the factorization-based methods, the corruption rate 0.2, the batch size 256, and the learning rate 0.001 with the Adam optimizer in TensorFlow[3] and the early stop strategy. Moreover, we use the Logistic loss in the output layer and the sigmoid function in both the hidden layer and output layer, randomly sample $5|\mathcal{R}|$ un-interacted (user, item) pairs as negative feedback, and choose the best regularization coefficient from {0.001, 0.01, 0.1}. For our k-RNN, we follow k-NN via fixing $|\tilde{\mathcal{N}}_u^{\ell}| = 100$ for fair comparison. For the size of $\mathcal{N}_u^k$ later used in the expanded neighborhood $\tilde{\mathcal{N}}_u^{\ell}$ in our k-RNN, we search the size $|\mathcal{N}_u^k| \in \{100, 200, 300, \ldots, 1000\}$ via the NDCG@5 performance on the first copy of the validation data.

Notice that the evaluation protocol used by some deep learning based recommendation methods is different from that of the conventional recommendation methods. For example, in NeuMF [13], it only samples 100 unobserved items from $\mathcal{I}\backslash\mathcal{I}_u$ instead of all the items in $\mathcal{I}\backslash\mathcal{I}_u$ for each user u because of the prohibitive time cost. It thus may not be appropriate for evaluation on large datasets used in our experiments.

4.4. Main results

From the results in Tables 3 and 4, we can have the following observations:

- Our k-RNN achieves significantly better[4] performance than all the baseline methods on all the five ranking-oriented evaluation metrics across the two datasets, which clearly shows the effectiveness of our k-RNN. In particular, our k-RNN beats k-NN, which is believed to be benefited from the effect of the symmetric reciprocal neighborhood.
- Among the baseline methods, we can see that both k-NN(user) and k-NN(item) beat the best-performing model-based methods, i.e., BPR and LogMF, in most cases, which shows the superiority of the neighborhood-based methods in comparison with the model-based methods in modeling users' one-class behaviors.
- For the recommendation methods based on a pointwise preference assumption (i.e., LogMF) and a pairwise preference assumption (i.e., BPR), we can see that the performance of these two methods are very close. Notice that the pairwise preference learning mechanism in BPR is usually recognized to be better than the counter part, i.e., pointwise preference learning, for the studied OCCF problem. This means that either the pointwise preference assumption or the pairwise preference assumption has the potential to perform well with appropriate loss functions.
- For the two pointwise preference learning methods, i.e., PMF and LogMF, we can see that LogMF performs much better than

[3] https://www.tensorflow.org/.

[4] We conduct significance test via the MATLAB function ttest2 http://www.mathworks.com/help/stats/ttest2.html.

Table 3
Recommendation performance of our *k*-reciprocal nearest neighbors algorithm (*k*-RNN), a closely related algorithm *k*-NN [5] with item neighborhood and user neighborhood, i.e., *k*-NN(item) and *k*-NN(user), some competitive model-based algorithm including factored item similarity model (FISM) [3], collaborative denoising auto-encoders (CDAE) [26], restricted Boltzman machine (RBM) [9], probabilistic matrix factorization (PMF) [23], Logistic matrix factorization (LogMF) [10], and Bayesian personalized ranking (BPR) [11], as well as a basic popularity-based recommendation method PopRank on ML20M w.r.t. five commonly used ranking-oriented evaluation metrics. We have marked the significantly best results in bold (the *p*-value is smaller than 0.01).

Method	Pre@5	Rec@5	F1@5	NDCG@5	1-call@5
PopRank	0.1038 ± 0.0003	0.0532 ± 0.0004	0.0566 ± 0.0003	0.1157 ± 0.0002	0.3737 ± 0.0010
FISM	0.1438 ± 0.0006	0.0869 ± 0.0006	0.0885 ± 0.0005	0.1605 ± 0.0006	0.4989 ± 0.0015
CDAE	0.1282 ± 0.0020	0.0715 ± 0.0022	0.0750 ± 0.0017	0.1406 ± 0.0025	0.4545 ± 0.0058
RBM	0.1600 ± 0.0019	0.0894 ± 0.0017	0.0933 ± 0.0015	0.1785 ± 0.0018	0.5293 ± 0.0036
PMF	0.1447 ± 0.0027	0.0843 ± 0.0020	0.0878 ± 0.0020	0.1595 ± 0.0033	0.4972 ± 0.0079
LogMF	0.1748 ± 0.0007	0.1088 ± 0.0003	0.1093 ± 0.0003	0.1969 ± 0.0011	0.5698 ± 0.0016
BPR	0.1776 ± 0.0003	0.0984 ± 0.0002	0.1034 ± 0.0002	0.1966 ± 0.0007	0.5591 ± 0.0012
k-NN(item)	0.1750 ± 0.0005	0.0984 ± 0.0002	0.1023 ± 0.0003	0.1976 ± 0.0007	0.5528 ± 0.0013
k-NN(user)	0.1901 ± 0.0004	0.1128 ± 0.0003	0.1146 ± 0.0003	0.2158 ± 0.0005	0.5900 ± 0.0010
k-RNN	**0.1984** ± 0.0002	**0.1175** ± 0.0002	**0.1197** ± 0.0002	**0.2247** ± 0.0004	**0.6086** ± 0.0003

Table 4
Recommendation performance of our *k*-reciprocal nearest neighbors algorithm (*k*-RNN), a closely related algorithm *k*-NN [5] with item neighborhood and user neighborhood, i.e., *k*-NN(item) and *k*-NN(user), some competitive model-based algorithm including factored item similarity model (FISM) [3], collaborative denoising auto-encoders (CDAE) [26], restricted Boltzman machine (RBM) [9], probabilistic matrix factorization (PMF) [23], Logistic matrix factorization (LogMF) [10], and Bayesian personalized ranking (BPR) [11], as well as a basic popularity-based recommendation method PopRank on Netflix w.r.t. five commonly used ranking-oriented evaluation metrics. We have marked the significantly best results in bold (the *p*-value is smaller than 0.01).

Method	Pre@5	Rec@5	F1@5	NDCG@5	1-call@5
PopRank	0.0934 ± 0.0003	0.0255 ± 0.0007	0.0318 ± 0.0004	0.0969 ± 0.0001	0.3311 ± 0.0003
FISM	0.1580 ± 0.0016	0.0591 ± 0.0010	0.0678 ± 0.0008	0.1688 ± 0.0023	0.5174 ± 0.0050
CDAE	0.1550 ± 0.0012	0.0525 ± 0.0009	0.0623 ± 0.0009	0.1639 ± 0.0014	0.4967 ± 0.0031
RBM	0.1559 ± 0.0020	0.0526 ± 0.0011	0.0621 ± 0.0004	0.1660 ± 0.0018	0.5004 ± 0.0007
PMF	0.1558 ± 0.0024	0.0574 ± 0.0008	0.0666 ± 0.0007	0.1650 ± 0.0017	0.5120 ± 0.0042
LogMF	0.1617 ± 0.0007	0.0665 ± 0.0006	0.0727 ± 0.0008	0.1742 ± 0.0008	0.5282 ± 0.0025
BPR	0.1747 ± 0.0006	0.0644 ± 0.0006	0.0732 ± 0.0004	0.1866 ± 0.0013	0.5423 ± 0.0005
k-NN(item)	0.1808 ± 0.0004	0.0647 ± 0.0003	0.0742 ± 0.0002	0.1948 ± 0.0004	0.5466 ± 0.0009
k-NN(user)	0.1795 ± 0.0001	0.0668 ± 0.0002	0.0756 ± 0.0001	0.1934 ± 0.0002	0.5508 ± 0.0010
k-RNN	**0.1853** ± 0.0001	**0.0686** ± 0.0002	**0.0780** ± 0.0001	**0.1998** ± 0.0002	**0.5618** ± 0.0003

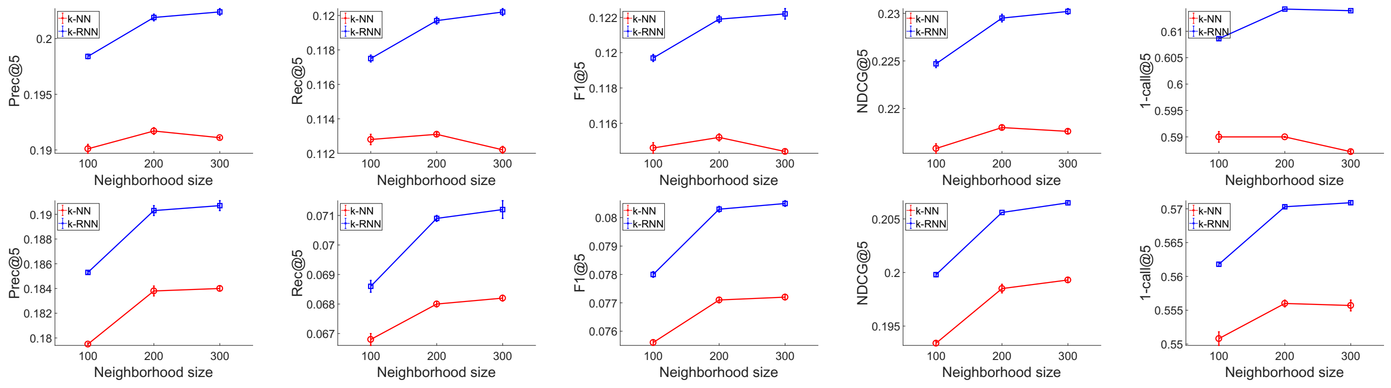


Fig. 5. Recommendation performance of *k*-NN and our *k*-RNN with different neighborhood sizes {100, 200, 300} on ML20M (top) and Netflix (bottom), respectively.

PMF, which clearly shows the effectiveness of the loss functions used in the two methods, i.e., the square loss in PMF and the Logistic loss in LogMF, which is the only difference between PMF and LogMF in our implementation.

- For the method that minimizes an energy function instead of a certain loss function (i.e., RBM) and the methods that learn similarities instead of using some predefined similarities (i.e., FISM and CDAE), we can see that their performance are close to that of PMF, which provides alternative ways for modeling one-class feedback.
- In comparison with the personalization methods (i.e., the model-based methods and the neighborhood-based methods), the non-personalization method PopRank performs worse in all cases, which is consistent with our common sense because each user's individual information is usually helpful in delivering top-*N* recommendation.

4.5. Impact of the number of neighbors

In order to study the impact of the size of the finally used neighborhood for preference prediction and item recommendation, i.e., the number of neighboring users in *k*-NN and our *k*-RNN, we change the size $|\mathcal{N}_u^k| \in \{100, 200, 300\}$ in *k*-NN and $|\tilde{\mathcal{N}}_u^\ell| \in \{100, 200, 300\}$ in our *k*-RNN. We report the top-*N* recommendation performance in Fig. 5. We can have the following observations:

- The performance of our *k*-RNN is much better than *k*-NN on both datasets when the corresponding neighborhood size is the same, i.e., both are 100, 200 or 300. This results again show the superiority of our k-RNN, in particular of the usefulness of the k-reciprocal nearest neighbors.

- Using a relatively large number of neighboring users, e.g., 200 or 300, both k-NN and our k-RNN perform better in most cases, where the improvement is more significant for our k-RNN. This is because that a proper larger number of neighbors will usually provide more information in the recommendation process. Moreover, the improvement from k-RNN with 200 to k-RNN with 300 is smaller than that from 100 to 200, which indicates that more neighbors brings less improvement. This is reasonable and consistent with previous studies on neighborhood-based methods.

4.6. Time cost

In this subsection, we study the time cost of our k-RNN and the most closely related algorithm k-NN. We conduct the experiments on a Windows machine with Intel(R) Core(TM) i7-3770 CPU @ 3.40 GHz (1-CPU/4-core)/16GB RAM. In particular, (i) on ML20M, k-NN with $|\mathcal{N}_u^k| = 100$ and k-RNN with $|\tilde{\mathcal{N}}_u^\ell| = 100$ take 14.36 h and 14.27 h, respectively; and (ii) on Netflix, they take 2.95 h and 2.96 h, respectively. From the above results, we can see that the time cost are consistent with the analysis in Section 3.4.3, i.e., the time complexity of k-NN and k-RNN are the same. We have also conducted additional experiments with larger sizes of $\mathcal{N}_u^k$ and $\tilde{\mathcal{N}}_u^\ell$, and find that the results are similar.

5. Conclusions and future work

In this paper, we propose a novel neighborhood-based collaborative filtering algorithm, i.e., k-reciprocal nearest neighbors algorithm (k-RNN), for a recent and important top-N recommendation problem called one-class collaborative filtering (OCCF). In particular, we propose to construct a symmetric reciprocal neighborhood for a better and stronger neighboring structure, which is then embedded in the prediction rule of a traditional k-NN algorithm. Notice that the main difference between k-NN and our k-RNN is the symmetric neighborhood used in preference prediction, which will not cause additional time cost (i.e., their time complexity is the same). We then study the effectiveness of the reciprocal neighborhood in comparison with the asymmetric neighborhood as well as the impact of the size of the neighborhood, and find that our k-RNN is a very promising recommendation algorithm. The experimental results and observations also give us some practical guidance in developing and deploying neighborhood-based recommendation algorithms.

For future works, we are interested in generalizing the concept of the reciprocal neighborhood from neighborhood-based collaborative filtering methods to model-based and deep learning based methods [26,27]. For example, we may explore the effectiveness of the mined reciprocal neighborhood in correlation and preference learning [26], and feed the knowledge of the symmetric neighborhood as an additional source of information to the layered neural networks for better recommendation [27].

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

We thank the handling editors and reviewers for their efforts and constructive expert comments, Mr. Yongxin Ni for his kind assistance in empirical studies, and the support of National Natural Science Foundation of China Nos. 61872249, 61836005 and 61672358. Weike Pan and Zhong Ming are the corresponding authors for this work.

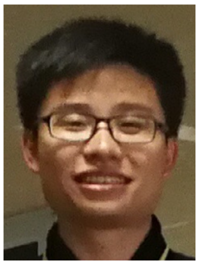
References

[1] F. Ricci, L. Rokach, B. Shapira, Recommender Systems Handbook, 2nd ed., Springer, 2015.
[2] R. Pan, Y. Zhou, B. Cao, N.N. Liu, R.M. Lukose, M. Scholz, Q. Yang, One-class collaborative filtering, in: Proceedings of the 8th IEEE International Conference on Data Mining, ICDM '08, 2008, pp. 502–511.
[3] S. Kabbur, X. Ning, G. Karypis, FISM: factored item similarity models for top-N recommender systems, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '13, 2013, pp. 659–667.
[4] E. Christakopoulou, G. Karypis, Local latent space models for top-N recommendation, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '18, 2018, pp. 1235–1243.
[5] M. Deshpande, G. Karypis, Item-based top-N recommendation algorithms, ACM Trans. Inf. Syst. 22 (1) (2004) 143–177.
[6] D. Goldberg, D. Nichols, B.M. Oki, D. Terry, Using collaborative filtering to weave an information tapestry, Commun. ACM 35 (12) (1992) 61–70.
[7] J. Gou, W. Qiu, Z. Yi, Y. Xu, Q. Mao, Y. Zhan, A local mean representation-based k-nearest neighbor classifier, ACM Trans. Intell. Syst. Technol. 10 (3) (2019) 29:1–29:25.
[8] D. Qin, S. Gammeter, L. Bossard, T. Quack, L.J.V. Gool, Hello neighbor: accurate object retrieval with k-reciprocal nearest neighbors, in: Proceedings the 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR '11, 2011, pp. 777–784.
[9] M. Jahrer, A. Töscher, Collaborative filtering ensemble for ranking, in: Proceedings of KDD Cup 2011 Competition, 2012, pp. 153–167.
[10] C.C. Johnson, Logistic matrix factorization for implicit feedback data, in: Proceedings of NeurIPS 2014 Workshop on Distributed Machine Learning and Matrix Computations, 2014.
[11] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, BPR: Bayesian personalized ranking from implicit feedback, in: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, UAI '09, 2009, pp. 452–461.
[12] Z. Wang, Y. Guo, B. Du, Matrix completion with preference ranking for top-N recommendation, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI '18, 2018, pp. 3585–3591.
[13] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proceedings of the 26th International Conference on World Wide Web, WWW '17, 2017, pp. 173–182.
[14] J.-P. Benzcri, Construction d'une classification ascendante hirarchique par la recherche en chaine des voisins rciproques, Cahiers l-Anal. Donnes 7 (2) (1982) 209–218.
[15] A. Lelu, Analyse en composantes locales et graphes de similarit entre textes, in: Procédure de the 7es Journees internationales d'Analyse statistique des Donnees Textuelles, JADT '04, 2004, pp. 737–742.
[16] A. Delvinioti, H. Jégou, L. Amsaleg, M.E. Houle, Image retrieval with reciprocal and shared nearest neighbors, in: Proceedings of the 9th International Conference on Computer Vision Theory and Applications, VISAPP '14, 2014, pp. 321–328.
[17] Z. Zhong, L. Zheng, D. Cao, S. Li, Re-ranking person re-identification with k-reciprocal encoding, in: Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '17, 2017, pp. 3652–3661.
[18] J. Gou, W. Qiu, Z. Yi, X. Shen, Y. Zhan, W. Ou, Locality constrained representation-based k-nearest neighbor classification, Knowl.-Based Syst. 167 (2019) 38–52.
[19] J. Gou, H. Ma, W. Ou, S. Zeng, Y. Rao, H. Yang, A generalized mean distance-based k-nearest neighbor classifier, Expert Syst. Appl. 115 (2019) 356–372.
[20] F.M. Harper, J.A. Konstan, The movielens datasets: history and context, ACM Trans. Interact. Intell. Syst. 5 (4) (2015) 19:1–19:19.
[21] W. Pan, L. Chen, Group Bayesian personalized ranking with rich interactions for one-class collaborative filtering, Neurocomputing 207 (2016) 501–510.
[22] D. Valcarce, A. Bellogín, J. Parapar, P. Castells, On the robustness and discriminative power of information retrieval metrics for top-N recommendation, in: Proceedings of the 12th ACM Conference on Recommender Systems, RecSys '18, 2018, pp. 260–268.
[23] R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, in: Annual Conference on Neural Information Processing Systems, NeurIPS '07, 2007, pp. 1257–1264.
[24] K. Georgiev, P. Nakov, A non-IID framework for collaborative filtering with restricted Boltzmann machines, in: Proceedings of the 30th International Conference on Machine Learning, ICML '13, 2013, pp. 1148–1156.
[25] G.E. Hinton, A practical guide to training restricted Boltzmann machines, in: Neural Networks: Tricks of the Trade, 2nd ed., Springer, 2012, pp. 599–619.
[26] Y. Wu, C. DuBois, A.X. Zheng, M. Ester, Collaborative denoising auto-encoders for top-N recommender systems, in: Proceedings of the 9th ACM International Conference on Web Search and Data Mining, WSDM '16, 2016, pp. 153–162.

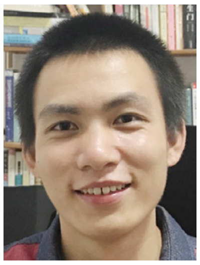
[27] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, G. Sun, xDeepFM: combining explicit and implicit feature interactions for recommender systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018, pp. 1754–1763.

Wei Cai is currently a final-year undergraduate student in the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include recommender systems and machine learning. He has won Bronze Medal of the 2018 China Collegiate Programming Contest, Qinhuangdao Site and Meritorious Winner of the Mathematical Contest in Modeling.

Weike Pan received the Ph.D. degree in Computer Science and Engineering from the Hong Kong University of Science and Technology, Kowloon, Hong Kong, China, in 2012. He is currently an associate professor with the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include transfer learning, recommender systems and machine learning. He has been active in professional services. He has served as an editorial board member of Neurocomputing, a co-guest editor of a special issue on big data of IEEE Intelligent Systems (2015–2016), an information officer of ACM Transactions on Intelligent Systems and Technology (2009–2015), and journal reviewer and conference/workshop PC member for dozens of journals, conferences and workshops.



Jixiong Liu received the B.S. degree in Computer Science and Technology from the Shenzhen University, Shenzhen, China, in 2018. He is currently a master student in the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include recommender systems and deep learning.



Zixiang Chen received the B.S. degree in Software Engineering from the Ludong University, Yantai, China, in 2016. He is currently a master student in the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include recommender systems and deep learning.

Zhong Ming received the Ph.D. degree in Computer Science and Technology from the Sun Yat-Sen University, Guangzhou, China, in 2003. He is currently a professor with the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include software engineering and web intelligence.