



User structural information in priority-based ranking for top-N recommendation

Mohammad Majid Fayezi¹ · Alireza Hashemi Golpayegani¹

Received: 11 May 2022 / Revised: 6 December 2022 / Accepted: 11 December 2022 / Published online: 6 January 2023
© The Author(s), under exclusive licence to Springer Nature Switzerland AG 2023

Abstract

The recommender system is a set of data recovery tools and techniques used to recommend items to users based on their selection. To improve the accuracy of the recommendation, the use of additional information (e.g., social information, trust, item tags, etc.) in addition to user-item ranking data has been an active area of research for the past decade.

In this paper, we present a new method for recommending top-N items, which uses structural information and trust among users within the social network and extracts the implicit connections between users and uses them in the item recommendation process. The proposed method has seven main steps: (i) extract items liked by neighbors, (ii) constructing item features for neighbors, (iii) extract embedding trust features for neighbors, (iv) create user-feature matrix, (v) calculate user's priority, (vi) calculate item's priority and finally, (vii) recommend top-N items. We implement the proposed method with three datasets for recommendations. We compare our results with some advanced ranking methods and observe that the accuracy of our method for all users and cold-start users improves. Our method can also create more items for cold-start users in the list of recommended items.

Keywords Recommendation system · Social network · Random walk · Community aware · Item ranking · Node embedding · Trust features · Tag set · Rating data

1 Introduction

Recommender systems suggest the most appropriate items (data, information, products, etc.) to users by analyzing their behavior. These systems are the approaches that have been proposed to overcome the problems occurred by the large and growing volume of information and help users get closer to their goal faster among huge amounts of information. Recommending systems are used in various fields, including offering products in business (such as Flipkart, Amazon, etc.) (Sarwar et al. 2002; Linden et al. 2003), recommending music and movies (such as Youtube, Lastfm, etc.) (Covington et al. 2016), recommending scientific articles (such as researchgate, google scholar, etc.) (Agarwal et al. 2005).

One of the most important issues is predicting the rate for unknown items, which is known as rating prediction. Also in the Top-N recommendation, best N item of a ranked list are offered to users. There are several types of recommendation systems, including:

(1) Collaborative filtering (CF), which uses item ranking information (Agarwal et al. 2005). In collaborative filtering method, suggestions are presented based on the selections of users who have similar behavior to the current user. More simply, the CF method is based on this assumption that users who have similar opinion about some items (movies, photos, music, etc.) have similar opinions about other items. (2) Content-based method: In this method, it uses the features of items for recommendation (Agarwal et al. 2005). It uses metadata such as genre, actor producer, and musician to describe movie or music items, etc.

(3) Hybrid method: In this method, both item ranking information and item features are used simultaneously (Burke 2002, 2007). The collaborative filtering suffers from the following disadvantages: (1) Sparsity of ranking data. This means that ranking data for a few items are created by

✉ Mohammad Majid Fayezi
majidfayezi@yahoo.com

Alireza Hashemi Golpayegani
sa.hashemi@aut.ac.ir

¹ Computer Engineering and Information Technology
Department, Amirkabir University of Technology, Tehran,
Iran

users and results in decreasing the accuracy of the recommendations. Various methods have been proposed to solve this problem. (2) The second problem concerns the issue of scalability. If the size of the ranking data becomes large, the time required to perform item recommendations will be high. There are several ways to solve this problem (Xue, et al. 2005), Wu et al. (2016), Bell and Koren (2007), (Koren 2010). (3) The third problem is related to the cold-start users. There is not enough background information for cold-start users. Therefore, providing suggestions for these users is challenging (Son 2016). Various methods have been proposed to address this challenge so far such as the use of trust and social information (Zhao et al. 2015), Guo (2013), (Lin et al. 2013).

In (Banerjee et al. 2021) for the problem of sparsity of ranking data and the problem of cold-start, a Top-N item recommendation technique is presented using user social network and item tags information as external information. In (Banerjee et al. 2021), to extract the features of neighbors, it uses items ranked by neighbors and obtains a Neighbors feature matrix. According to this matrix, it ranks neighbors, items and finally Top-N recommendations provides to the target user. One of the main challenges in this approach is defining the neighborhood for the target user. In (Banerjee et al. 2021), only direct neighbors are considered. Considering farther neighbors for the target user can improve ranking for users and items and recommending items become more accurately.

In addition, adding structural information with considering more distant neighbors will provide better external information for cold-start users, thus this problem will be better addressed. Various methods have been proposed to consider structural information. Due to the sparsity of ranking data, random walk-based methods have been widely used. One of these methods is the CARE method (Keikha et al. 2018). In this method, in addition to local neighborhood information, it uses the information of network communities to extract global structural information. It uses random walks to create paths between nodes, and extract neighborhood information. When the target node has not enough neighbors, a random node from its community is selected and placed in the path.

The CARE method does not make a difference in the importance of direct neighbors with co-community neighbors which leads to neglect the effective information. In other words, the impact of neighbor's node information should be greater than the impact of co-community's node information. Thus, in our paper, a new distance-based method as the importance measure for nodes is used in the recommendation process. The proposed method consists of 7 steps.

(1) Extract items that have been liked by neighbors.

(2) Extract features of each item and consider them to neighbors who ranked that item.

(3) Extract structural features for neighbors: According to the improved CARE method for each neighbor in the social network, we extract its structural features. These features are obtained according to how users communicate with each other (such as neighbor or co-community).

(4) Create neighbor's feature matrix: This matrix is obtained by combining item features and structural features for neighbors.

(5) Create neighbor priority: a distance-based score method on the neighbor's feature matrix is performed to rank neighbors.

(6) Create item priority: Based on the ranking of neighbors, unknown items are prioritized for the target user.

(7) Provide Top-N ranked items for the target user, according to item priority.

The key contributions of this paper can be summarized as follows:

- Adding structural information by considering farther and co-community neighbors for the target user instead of just direct neighbors can lead to the extraction of valuable information and better address the cold-start user problem.
- Determine the importance coefficients for structural information driven from farther and co-community neighbors in the recommendation process. The coefficients prevent the loss of effective information
- Considering the use of external information, we show that the cold-start scenario improves.
- We present an improved CARE algorithm that distinguishes between direct and farther neighbor structural information
- Provide a new distance-based method as the importance measure for nodes is used in the recommendation process.

The rest of the paper is organized as follows. In Sect. 2, we will review the related works. In Sect. 3, we describe our method and explain its architecture and the corresponding analyses. Section 4 includes experimental evaluations of our proposed method, the experimental setup, evaluation metric, and results and discussion are explained in this section. Finally, in Sect. 5, the conclusion of our work is given.

2 Related works

In the following, the related works are explained.

2.1 Network embedding

There is a significant increase in growing online social networks and the number of their users. Useful information

can be exploited from social networks with analyzing their structure and content. Machine learning methods are used to extract useful features from social networks for some tasks like classification (Tsoumakas and Katakis 2006), Sen et al. (2008), (Getoor and Taskar 2007), recommendation (Fouss et al. 2007; Backstrom and Leskovec 2011) and link prediction (Yang et al. 2011), Vazquez et al. (2003), Radivojac et al. (2013), (Liben-Nowell and Kleinberg 2007). These learning methods can be either supervised or unsupervised. Supervised learning algorithms can better extract features for a particular task on social networks, but they are not suitable for large networks. On the other hand, unsupervised methods can manage the scalability of feature learning methods. However, the extracted features show low accuracy in network analysis tasks. They are very general to give useful information for a particular task (Yan et al. 2006), Tenenbaum et al. 2000, Roweis and Saul 2000, Pennington et al. 2014, Mikolov et al. 2013, Bengio et al. 2013 (Belkin and Niyogi 2001).

Network embedding, as an unsupervised learning, seeks to extract useful information by representing nodes with low dimension and learning the social relationships of network nodes in a low dimension space to maintain network structure. These vector representations can be used in the analysis of various social network tasks such as classification, recommendation (Bhagat et al. 2011) and link prediction (Liben-Nowell and Kleinberg 2007). Some classical network embedding methods use special dependency graph vectors as feature vectors (Belkin and Niyogi 2001; Tenenbaum et al. 2000) (Cox and Cox 2008; Roweis and Saul 2000). Graph factorization is another technique used for network embedding (Ahmed et al. 2013). The mentioned approaches suffer from scalability for large social networks. In recent years, deep learning as an unsupervised method has been widely used in natural language processing, a detailed description of this research can be found in Bengio et al. (2013).

There is also a lot of research that has used in-depth learning for social network embedding (Perozzi et al. 2014; Tang et al. 2015; Grover and Leskovec 2016) (Wang et al. 2017). Network embedding methods try to represent graph nodes with some useful feature vectors. Deepwalk (Perozzi et al. 2014), LINE (Tang et al. 2015) and Node2vec (Grover and Leskovec 2016) are the most important methods proposed in recent years. Although these methods perform well compared to other graph representation methods such as spectral clustering, they attempt to extract only local structural information from each node and then use them to learn the final representation of the node. Communities, however, have important structural information that are neglected by these methods (Wang et al. 2016). The structure of society provides additional information to display nodes. For two nodes in a community, the similarity of representation of these two nodes must increase, even if their relationship is weak due to

data sparsity. Thus, combining community structure in network embedding can provide effective and rich information to solve data sparsity problems and, in addition, make node representations become more distinctive (Wang et al. 2017). As mentioned, network embedding refers to the approach of learning embedded features with low dimension for nodes or links in a network. The basic principle is to learn encryption for network nodes in such a way that the similarity in the embedded space shows the similarity in the network. The applications of embedding nodes can be different and they are used for different types of graphs. The advantage of node embedding as a technique is that it does not require feature engineering by specialists in the field. One of the algorithms for embedding the network is deepwalk algorithm.

One of the new ways for recommending items to the user is the method described in Banerjee et al. (2021). This article uses three datasets, user-item ranking data, social network information between users, and item-feature set to provide recommendations to users. One of the challenges in this article is the lack of use of embedded and structural information within the user network. In (Banerjee et al. 2021), the neighbors are used to provide recommendations. These neighbors are directly (with one edge) connected to the target node, while the more distant neighbors are ignored in Banerjee et al. (2021). Using farther neighbors for a node can provide valuable information about the similarities between nodes and can improve the item recommendation system.

2.2 Deepwalk

In DEEPWALK, deep learning (unsupervised feature learning) was first used to learn the social representation of graph nodes by modeling the flow of random short walks. This algorithm learns to represent embedded features that encode social relations in a continuous vector space with a relatively small number of dimensions. Deepwalk has surpassed other latent representation methods in creating social dimensions, especially when labeled nodes are scarce. The representations learned by Deepwalk make strong predictive performance possible with very simple linear models. In addition, the resulting representations are general and can be combined with any classification method (Backstrom and Leskovec 2011). Deepwalk is an online algorithm that can be paralleled.

Deepwalk considers the problem of classifying members of a social network into one or more categories. Let $G = (V, E)$ and $G_L = (V, E, X, Y)$ be a partially labeled social network with input characteristics $X \in \mathbb{R}^{V \times S}$, such that S is the size of the feature space for each feature vector, and $Y = \mathbb{R}^{|V| \times |Y|}$. Given that Y is a set of possible tags. Deepwalk's goal is to learn $X_E \in \mathbb{R}^{|V| \times d}$, where d is a small number of dimensions. These dimensional representations are distributed, meaning that each social phenomenon

Table 1 Symbols used in this article

Symbols	Description
U, I, F	Set of users, items and features
A, R, M, UF	Adjacency, rating, Item-feature and User-feature matrix
$I_{u_t}^N$	Set of items rated by the neighbor users of u_t
N_t	Neighbors of the target user t
\hat{f}_i	i^{th} proximity feature
\overline{UF}	Normalized User-feature matrix
P	Priority array for neighbors users

is expressed by a subset of dimensions, and each dimension refers to a subset of the social concepts expressed in the network (Backstrom and Leskovec 2011). This method satisfies these requirements by learning the representation of nodes from the flow of random short walks, using optimization techniques designed for language modeling (Backstrom and Leskovec 2011).

The Deepwalk algorithm consists of two main components. First a random walk generator, and second, an update method. The random walk generator takes a graph G and uniformly samples a random node v_i as the root of the random walk W_{v_i} . A walk is sampled from the neighbors of the last visited node to reach the maximum length (t). While the length of random walks is fixed in experiments, there is no limit to the length of random walks.

2.3 Skipgram

Skipgram is a language model that maximizes the probability of concurrency between words that appear within a w window in a sentence. This method approximates the conditional probability using the assumption of independence as follows:

$$P(\{v_{i-w}, \dots, v_{i+w}\} | v_i) = \prod_{j=i-w, j \neq i}^{i+w} P(v_j | v_i), \tag{1}$$

where $\Phi(v_i)$ is a representation feature vector for the v_i node. The purpose of the Skipgram model in Deepwalk is to extract the local structure around the v_i node, which is defined as the surrounding w node.

3 Proposed algorithm

The symbols user in this article is described in Table 1.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7
u_1	1	0	1	0	0	1	0
u_2	0	1	0	0	0	1	0
u_3	1	0	0	1	0	1	0
u_4	0	1	1	0	0	1	0
u_5	1	0	1	1	0	0	0
u_6	1	0	0	0	0	0	1
u_7	0	0	1	1	0	0	0
u_8	0	1	0	1	1	0	1

Fig. 1 User-item rating data

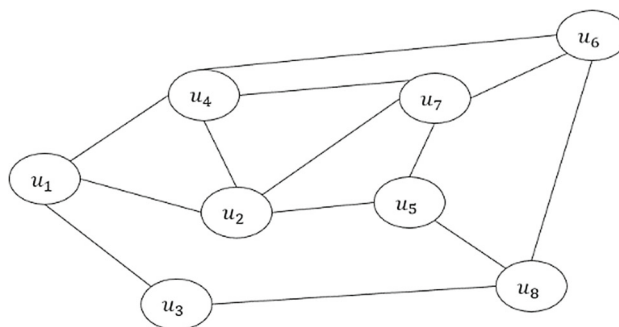


Fig. 2 Social network among users

3.1 Illustration with example

Here, we provide a simple example to demonstrate the proposed method. As in Banerjee et al. (2021), we consider a recommender system with 8 users and 7 items formed as $U = \{u_1, u_2, \dots, u_8\}$ and $I = \{i_1, i_2, \dots, i_7\}$. The user-item rating data matrix is formed as matrix R and presented in Fig. 1.

The social network among users is shown in Fig. 2. The items are described by the feature set $F = \{f_1, f_2, \dots, f_8\}$, and the item-feature matrix is displayed as M in Fig. 3. If we assume that the target user is u_5 , then $N = \{u_2, u_7, u_8\}$ and $I_{u_5}^N = \{i_2, i_3, i_4, i_5, i_6, i_7\}$. User-feature matrix named as UF for user u_5 is as follows.

$$M = \begin{matrix} & \begin{matrix} f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 \end{matrix} \\ \begin{matrix} i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \\ i_6 \\ i_7 \end{matrix} & \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} \end{matrix}$$

Fig. 3 Item-feature matrix

$$UF = \begin{matrix} & \begin{matrix} f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 \end{matrix} \\ \begin{matrix} u_2 \\ u_7 \\ u_8 \\ u_5 \end{matrix} & \begin{pmatrix} 2 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 \\ 2 & 1 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 2 & 2 & 2 & 2 & 2 & 1 & 0 \end{pmatrix} \end{matrix}$$

As can be seen, for each user among $N = \{u_2, u_7, u_8\}$, features of ranked items are placed. In the UF table, the similarity between the target user u_5 and users $N = \{u_2, u_7, u_8\}$ is through the degree of similarity of users' features which are obtained through the ranks given to the items.

Therefore, the proximity (embedded) features between u_5 and nodes in N are not considered. Nodes that are close to each other should achieve more similar proximity features to each other. So, the proximity features should be considered for each node in the UF matrix.

To obtain the proximity features, the improved CARE algorithm is used to create optimal features for each node based on the paths between nodes. So three features $\{f'_1, f'_2, f'_3\}$ are obtained for each node and the $UF_{proximity}$ matrix becomes as follows.

$$UF_{proximity} = \begin{matrix} & \begin{matrix} f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 & f'_1 & f'_2 & f'_3 \end{matrix} \\ \begin{matrix} u_2 \\ u_7 \\ u_8 \\ u_5 \end{matrix} & \begin{pmatrix} 2 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0.4091 & 1.1037 & -0.0255 \\ 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 0.4701 & 1.0332 & -0.3098 \\ 2 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 0.1918 & 1.1504 & -0.3163 \\ 1 & 2 & 2 & 2 & 2 & 2 & 1 & 0 & 0.2311 & 1.1855 & -0.0173 \end{pmatrix} \end{matrix}$$

The normalized $UF_{proximity}$ matrix is as follows:

$$\overline{UF}_{proximity} = \begin{matrix} & \begin{matrix} f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 & f'_1 & f'_2 & f'_3 \end{matrix} \\ \begin{matrix} u_2 \\ u_7 \\ u_8 \\ u_5 \end{matrix} & \begin{pmatrix} 0.2097 & 0.1048 & 0.1048 & 0.1048 & 0 & 0.1048 & 0.1048 & 0.1048 & 0.0428 & 0.1157 & -0.002 \\ 0.0925 & 0.0925 & 0.0925 & 0.0925 & 0.1850 & 0.0925 & 0.0925 & 0.0925 & 0.0434 & 0.0955 & -0.0286 \\ 0.1201 & 0.0600 & 0.1201 & 0.1201 & 0.1201 & 0.1201 & 0.1201 & 0.1201 & 0.0115 & 0.0690 & -0.0189 \\ 0.0744 & 0.1489 & 0.1489 & 0.1489 & 0.1489 & 0.1489 & 0.0744 & 0 & 0.0172 & 0.0882 & -0.0012 \end{pmatrix} \end{matrix}$$

Now the distance between profile vectors of user u_5 and his neighbor user profile vectors is calculated and the distance array D is obtained.

The remaining steps are done as in Banerjee et al. (2021). As can be seen from the distance vector D , the distance between the nodes is calculated by considering their proximity features, and makes the similarity between the nodes calculated more accurate than (Banerjee et al. 2021).

3.2 Proposed idea

To address these challenges, in this paper, we want to add structural features among nodes to the recommending process. Adding these features will give us the following benefits:

- Addressing the cold-start problem due to the use of structural information among users. In the proposed idea, using indirect neighboring nodes information, it can provide better suggestions when we have incomplete information about the target node compared with the method (Banerjee et al. 2021).
- Increasing the accuracy of predictions and recommendations. By increasing the amount of information used to recommend the item to the target user, better and more accurate suggestions can be made.
- Adding trust information to the model. Information such as the number of communication between neighboring nodes is added to the model as trust information to increase the accuracy of recommendations.

In a recommender system, there are basically three entities: users, items, and user-item interactions in terms of ranking. We represent the user set with U , i.e., $U = \{u_1, u_2, \dots, u_{n1}\}$ and the set of items with I , i.e $I = \{i_1, i_2, \dots, i_{n2}\}$. The number of users and items in the system are denoted by $n1$ and $n2$, respectively.

A user is interested in different items in terms of ranking on a specific ranking scale, for example $\{1, 2, 3, 4, 5\}$. Points can be displayed as a tuple (u_p, i_q, r_{pq}) , indicating that the user u_p has rated i_q item with the value of r_{pq} . Ratings given to all system users for all items is considered by matrix R . Same as (Banerjee et al. 2021), we assume three different types of information as follows:

(i) User-item ranking data;

- User-item ranking data are given as a rating matrix R . The contents of the matrix R are as follows.

$$R_{pq} = \begin{cases} x & \text{if user } u_p \text{ rates the item } i_q \text{ with rating value } x \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

(ii) Social network between users.

- Social network information among users is given by the adjacency matrix (A) for the social network $G(U, E)$ without direction and weight. Here, the set of vertices (users) denoted as $V(G) = U = \{u_1, u_2, \dots, u_{n1}\}$ and the set of edges as $E(G) = \{(u_p, u_q) | u_p \text{ and } u_q \text{ has social relation}\}$.

(iii) A set of item properties.

- The description of an item is generally provided with a set of features. $F = \{f_1, f_2, \dots, f_{n3}\}$ shows the set of properties of one item. Thus, the property set of items is represented by an item-feature matrix $M \in \mathbb{R}^{n2 \times n3}$.

In this work, we consider only binary feature vectors. Therefore, if the item i_q has the feature f_r , the (q, r) -th of the matrix M will be one, otherwise it will be 0. Our proposed method is mainly divided into seven general steps. An overview of our proposed method is given in Fig. 4. In the following, we provide a step-by-step explanation of our proposed method.

1. Items likes by neighbors:

We indicate the target user as u_t . Initially, from the social network G , the neighboring users u_t in G are listed in N_t . Then, for all users in N_t , their ranked items are identified and kept in a list. We show this list as $I_t^{N_t}$. Thus, visually $I_t^{N_t} = \{i_q | \exists u_p \in N_t \text{ and } R_{u_p i_q} \neq 0\}$. In step 1, a set of neighboring user items, namely $I_t^{N_t}$, was created from the user-item ranking data and the social network G .

2. Item features for neighbors:

In this step, from the set of neighboring user items and description of the existing items, the user-feature matrix is created for the neighboring users. This matrix records the number of times different features are rated by each user in the list N . In the last row of this matrix, we attach the same information to the target user. We represent this matrix with UF .

3. Extract embedding trust features for neighbors

In our proposed method, the improved CARE algorithm is used to capture the embedding features among users. In the improved CARE algorithm, a number of embedded features as trust information among users are extracted

for each user. In this algorithm, users that are closer to each other get more similar features and those that are placed farther away get more different features. Therefore, more connections between the nodes, causes more similar embedding features for users and vice versa. As can be seen in Fig. 5, the features extracted by the improved CARE algorithm are the trust information for each user. In section 3.2, the improved CARE algorithm is explained.

4. User-feature matrix

After extracting trust features for users and item features for neighbors (similar to Banerjee et al. (2021)), these features are combined as shown in Fig. 6. In the last row of user-feature matrix, we add features for target user.

5. User's priority

From the user-feature matrix calculated in the previous step, here we calculate the priority between the neighboring users and the target user. First, we normalize the user-feature matrix by dividing each entry by the sum of the corresponding rows. We represent the normalized user-feature matrix with \overline{UF} .

$$\overline{UF}_{P_j} = \frac{UF_{P_j}}{\sum_{k=1}^r UF_{P_k}}. \quad (3)$$

We consider each row of the normalized user-feature matrix as the corresponding user profile vector. Now, if we calculate the distance between the target user profile vector (V_{u_t}) and each of its neighbors, this value can be considered as a measure for the correlation between the target user and its neighbors.

Here, we consider L2 norm as the distance metric between two vectors. For each specific attribute, in the normalized user-feature matrix, if the value for the target user is zero, then we do not consider that attribute in the distance calculation.

Hence, the distance between the neighboring user profile vector (V_{u_p}) and the target user profile vector V_{u_t} is as follows:

$$D_{tp} = \sqrt{\sum_{r \in F_{u_t > 0}} (V_{u_t}[r] - V_{u_p}[r])^2}. \quad (4)$$

Using Eq. (4), the distance vectors of the neighboring user profiles with the target user profile are calculated and stored in the distance array D . Now, intuitively, if the distance becomes greater, then the priority should be less. In this paper, we calculate P_{u_p} as user's priority for user u_p in Eq. (5). P_{u_p} is calculated from user distance $D_{u_p}^k$.

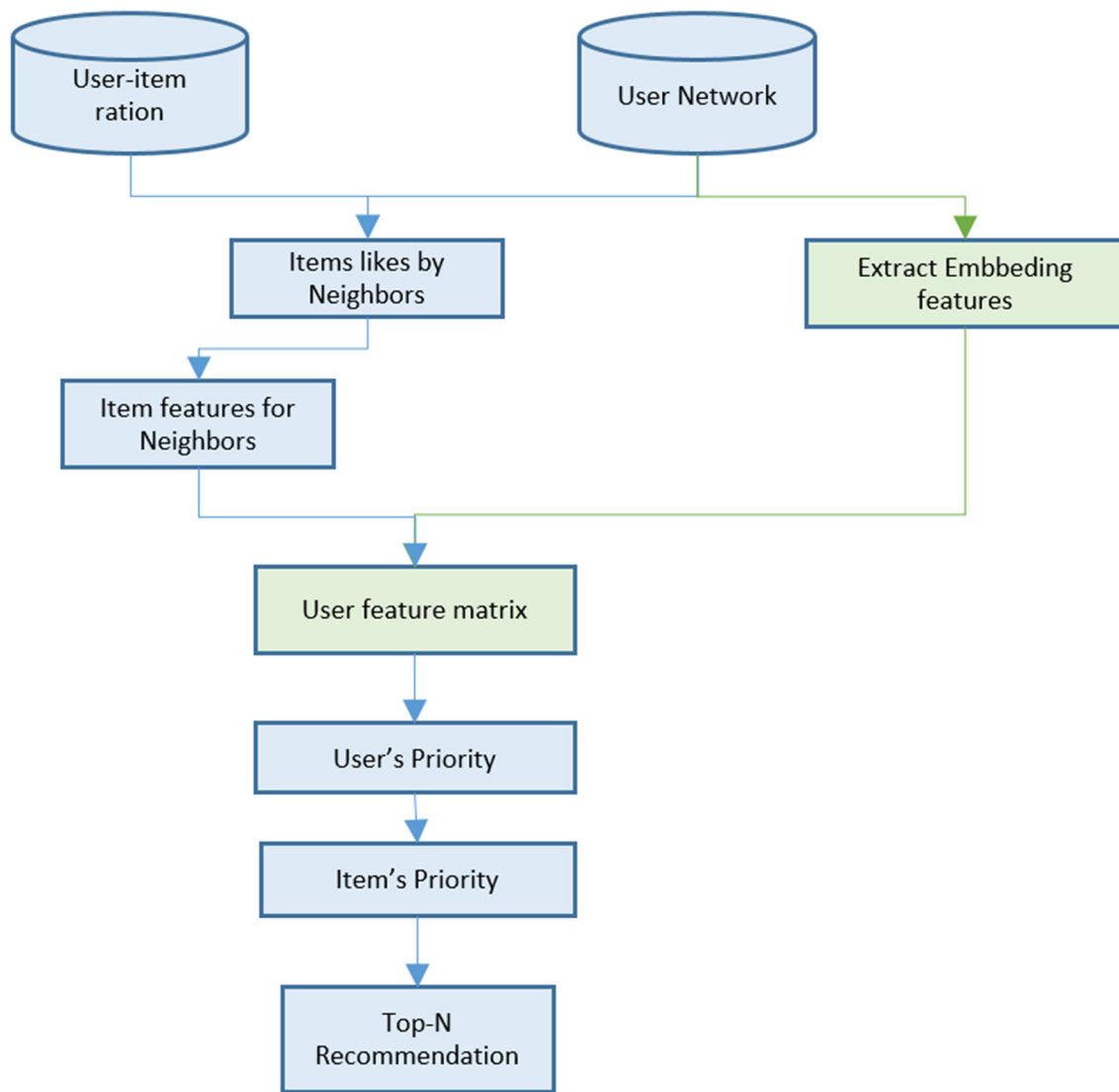


Fig. 4 Overview of our proposed method

$$P_{u_p} = \frac{1}{1 + D_{u_p}^k} \tag{5}$$

Here, P_{u_p} and D_{u_p} are the priority and distance of the neighboring user u_p , respectively, and k is a positive integer.

6. **Item’s priority**

In this step, we calculate the item’s priority for the recommendation based on the previously calculated user’s priority. Here, we calculate the item’s priority as the sum of the priorities of users ranked the corresponding item. Hence, the priority of the item i_q (denoted as P_{i_q}) can be determined by the following equation:

$$P_{i_q} = \sum_{u_p \in N, R_{pq} > 0} P_{u_p} \tag{6}$$

7. **Top-N recommendation**

In the last step, we recommend the items to the target user based on the calculated item’s priorities. We sort the items in descending order and recommend top-N items.

3.3 **Improved CARE algorithm**

In this algorithm, we use structural information (such as clustering) effectively in the network embedding process. The steps of improved CARE algorithm are as follows:

3.3.1 **Community detection**

We have used the Louvain method to maximize network modularity to identify communities (Morgan and Govender

Fig. 5 Embedding trust features for neighbors

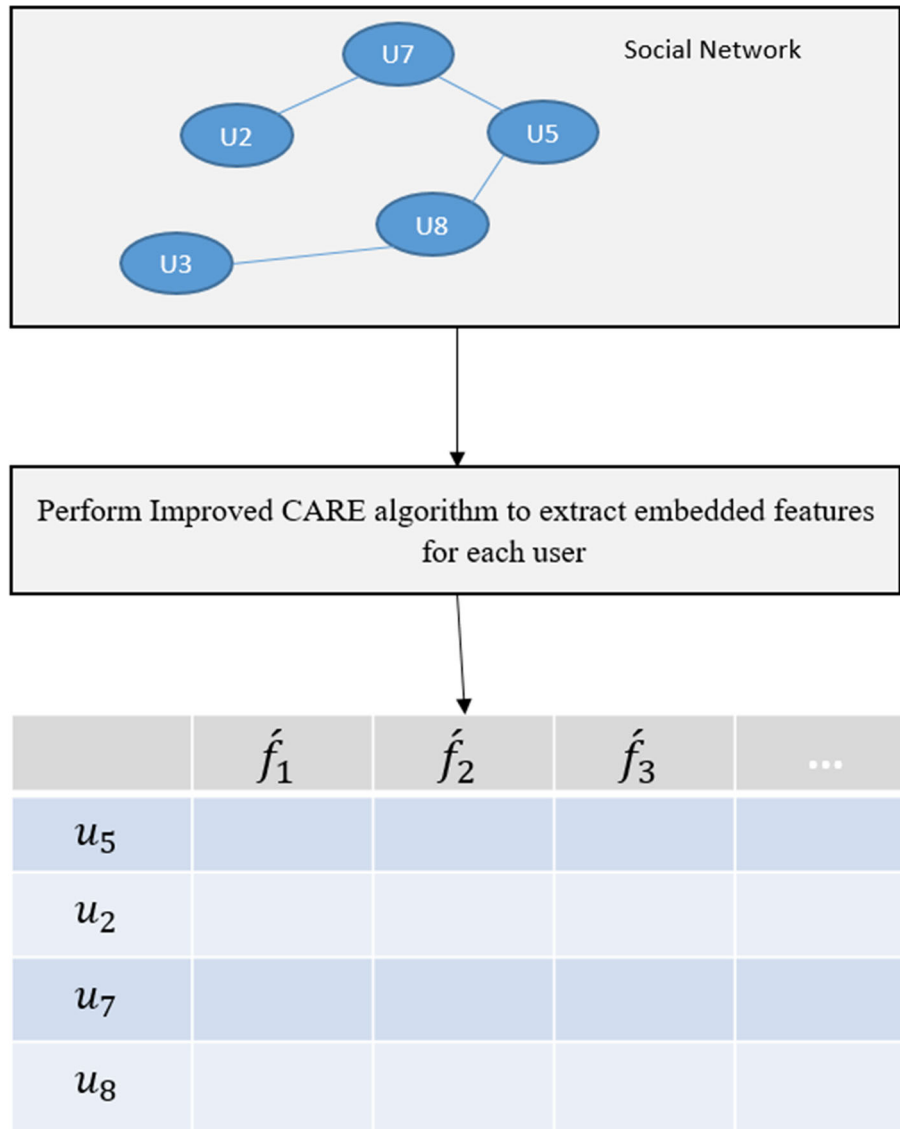
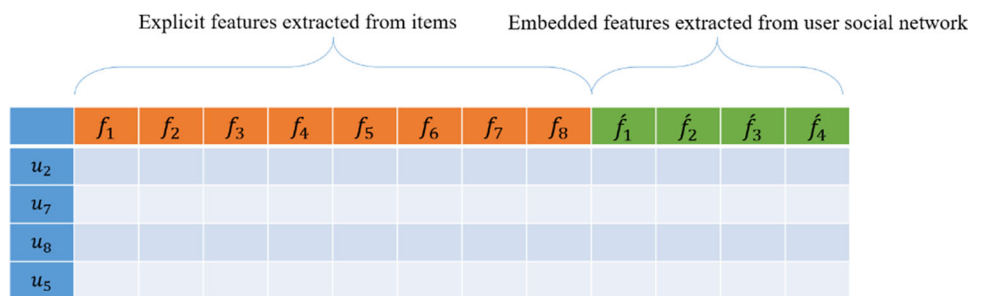


Fig. 6 User-feature matrix



2017). Modularity is a metric for comparing the density of edges within a community and the edges between communities. This is an optimization algorithm that first examines each node in a separate community. Then, a node is selected and the modularity of its connection to neighboring communities is calculated. Finally assigns the node to a community, where its modularity is maximized (Fig. 7).

3.3.2 Extract neighborhood structure

To extract the neighborhood structure of a node, we create customized random walks μ . Custom random walk starting from node v is indicated by v . Because a random walk is a path in a network. For example, we consider a custom random walk for node v , as $w_v^1, w_v^2, \dots, w_v^k$ such that w_v^k is a node that

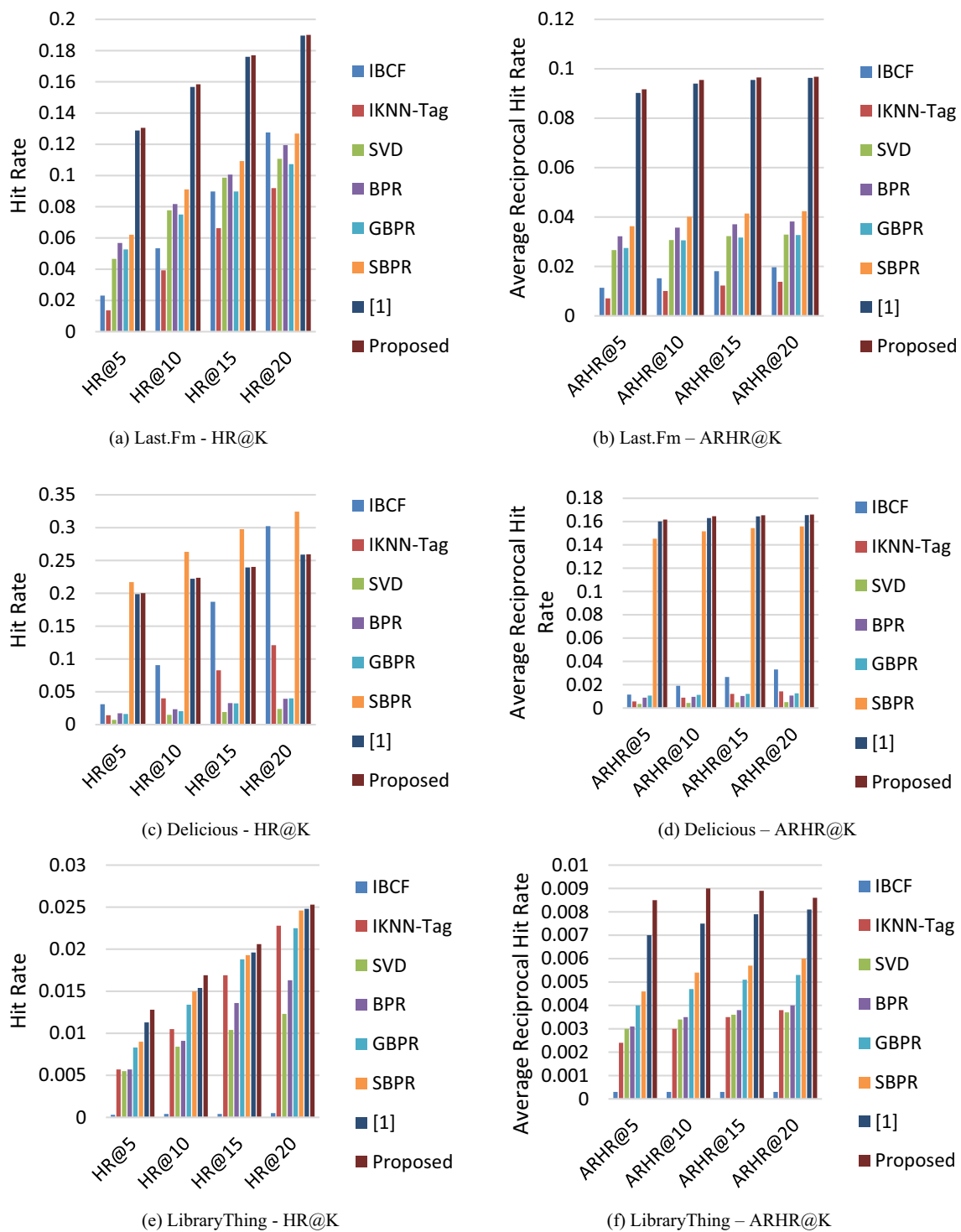


Fig. 7 Results for All-user. **a** Last.Fm—HR@K; **b** Last.Fm – ARHR@K; **c** Delicious – HR@K; **d** Delicious – ARHR@K; **e** LibraryThing – HR@K; **f** LibraryThing – ARHR@K

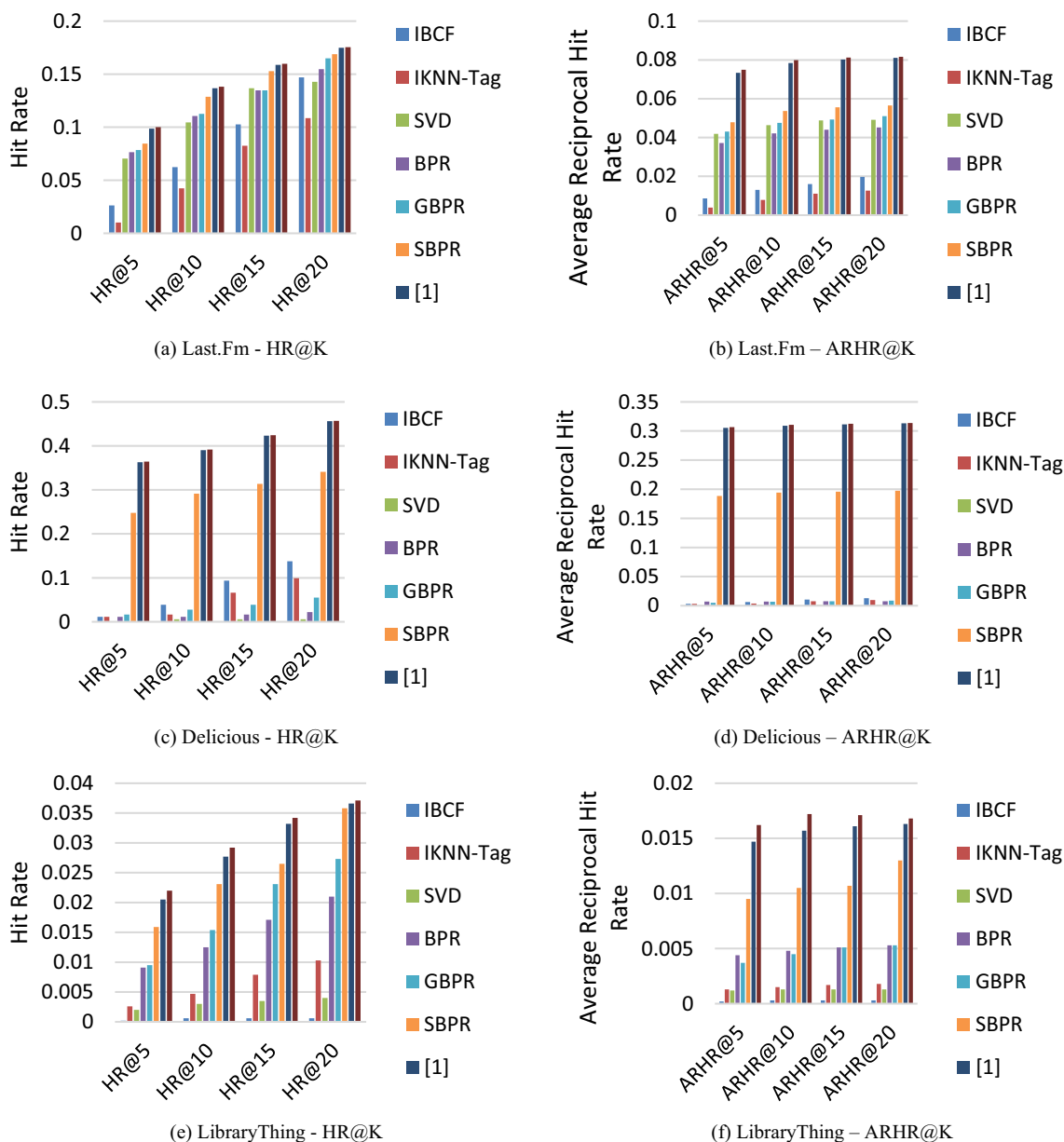


Fig. 8 Results for Cold-start users. **a** Last.Fm—HR@K; **b** Last.Fm – ARHR@K; **c** Delicious – HR@K; **d** Delicious – ARHR@K; **e** LibraryThing – HR@K; **f** LibraryThing – ARHR@K

is randomly selected from direct neighbors v or nodes placed in the same community. To create a custom random walk starting from node v , we first extract all its direct neighbors. Then, a random variable r between 0 and 1 is generated. If r is less than random variable α , we select a random node from direct neighbors; otherwise, we select a random node from nodes placed in the same community. This process continues until it reaches the predefined length l for the path (Fig. 8).

3.3.3 Skipgram

After generating random walks, we use the Skipgram model to learn graph node representations (Lin and Cohen 2010; Kondor and Lafferty 2002). Skipgram is a language model that maximizes the conditional probability of common words in a predefined window. As shown in the following equation:

$$\Pr(w|f(u)) = \max_f \prod_{j=i-w}^{i+w} pr(v_j|f(u))_{w = \{v_{i-w}, \dots, v_{i+w}\}}. \tag{7}$$

For each node in the graph, we repeat this maximization over all our custom random walks. We define a window w to move in one direction. Similar to the previous approaches, the assumption that conditional probabilities are independent is considered in Eq. (7). In addition, the softmax functions are used to approximate the probability distribution of Eq. (7):

$$\Pr(v_j|f(u)) = \frac{1}{1 + e^{-f(u)f(v_j)}}. \quad (8)$$

Stochastic gradient descent (SGD) is used to optimize the parameters, similar to the method proposed in Bottou (1991).

3.3.4 Shortest path algorithm

In graph theory, the problem of finding the shortest path is defined as the problem of finding the path between two vertices (or nodes) in such a way that the sum of the edge weights are minimized. For example, consider finding the fastest way from one place to another on the map; In this case, the vertices represent the places and the edges represent the parts of the path that are weighed according to the time required to travel them. The corresponding graph can be weighted or weightless (all edge weights are one).

The most important algorithms to find the shortest path are:

- Dijkstra algorithm that solves the problem of finding the shortest path between two vertices, from a single origin to a single destination.
- Bellman–Ford algorithm that solves the problem of finding the shortest path from a single origin where the weight of the edges can also be negative.
- Floyd–Warshall algorithm that solves the problem of finding the shortest path between two vertices.
- Johnson’s algorithm which solves the problem of finding the shortest path between two vertices and may work faster than Floyd–Warshall in scattered graphs.

In this paper, we use Dijkstra algorithm to solve this problem.

4 Proposed skipgram

Skipgram language model, which maximizes the conditional probability of common words in a predefined window, is defined according to Eq. (9).

$$\Pr(w|f(u)) = \max_f \prod_{j=i-w}^{i+w} pr(v_j|f(u))w = \{v_{i-w}, \dots, v_{i+w}\}. \quad (9)$$

$$\Pr(v_j|f(u)) = \frac{1}{1 + e^{-f(u)f(v_j) \frac{1}{\text{shortestPath}(u, v_j)}}}, \quad (10)$$

where $\text{shortestPath}(u, v_j)$ represents the ortest distance between two nodes u and v_j . The equation $\frac{1}{\text{shortestPath}(u, v_j)}$ is considered as a similarity measure for two nodes. Thus, for neighbor nodes, this shortest path takes a value of one and also $\frac{1}{\text{shortestPath}(u, v_j)}$ takes value one. On the other hand, for nodes in the same community, if their shortest distance is two, their similarity coefficient is $\frac{1}{2} = 0.5$, and so on. By changing softmax to Eq. (10), neighbor nodes are considered more importance than same community nodes, and thus we can produce a better representation model than the original CARE method for network nodes. Also, similar to the CARE method (Cox and Cox 2008), stochastic gradient descent (SGD) is used to optimize the parameters.

5 Experimental evaluation

In this section, we present the experimental results by comparing the proposed method with the existing recommendation algorithms. We evaluate our model based on three ranking datasets, which contain social tag information and items. In the experiments, we use two publicly available datasets called Last.fm and delicious, which contain user–item interaction data such as implicit ranking, social networking, and item social tagging information. We also use LibraryThing dataset that contains only social network rankings and information. We have collected tag data for items in the LibraryThing dataset from <https://www.librarything.com/>. The collected data are preprocessed by removing tags that are not meaningful and used more often. Also, we remove tags that are associated with less than twenty items. The basic statistics of the ranking dataset and additional information are given in Tables 1 and 2, respectively.

5.1 Experimental setup

To analyze the impact of combining social data and tags, we select users who have at least one social neighbor, and select items that have tag information. For analysis, we consider the leave-one mechanism. In this mechanism, for example, for each user, a ranked item is removed from the training set and placed in the test set, and the system performance is evaluated based on the presence of the test item in the list of recommended items. The following is a brief description of the methods that are compared to our proposed method.

- Item-Based collaborative Filtering (IBCF) (Deshpande and Karypis 2004): In this algorithm, item similarity is calculated using Pearson correlation similarity from the

Table 2 Statistics of the rating dataset

Dataset	#Users	#Items	#Ratings	Density	Cuser	Ritem
Last.fm	1892	12519	70484	0.0029	37.25	5.63
Delicious	1792	31195	301495	0.0054	168.24	9.66
LibraryThing	23499	45587	606687	0.0006	25.82	13.31

R ranking matrix. For top- K similar items, the similarity score are stored for each item, and for the unrated user-items pairs the rating score is predicted. So, the top- N items are recommended for each test user, based on the predicted score.

- **Item-KNN-tag:** Similar to IBCF, it uses the item-feature M matrix to obtain item similarity scores. This method does not follow the principle of collaborative filtering. It uses content information of items and recommends the most similar items.
- **Matrix Factorization (SVD) (Funk 2006):** Here, the R ranking matrix is broken down into two embedded feature matrices of users and items, and the ranking is predicted as a multiplication of the embedded feature vectors of the user and items. Finally, based on the predicted rankings, top- N items are recommended for each user.
- **Bayesian personalized ranking-based matrix factorization (BPRMF) (Rendle et al. 2012):** Using matrix factorization, the embedded features are learned for a user using a pairwise ranking of purchased and non-purchased items.
- **Group preference-based Bayesian personalized ranking (GBPR) (Pan and Chen 2013):** Similar to BPRMF, here the ranking is based on the target user and a random group of users.
- **Social Bayesian personalized ranking (SBPR) (Zhao et al. 2014):** Here, feature matrices are learned through the triple ranking of items (self-purchased, purchased by social neighbors, other items that not purchased) for each user.

5.2 Evaluation metric

The performance of our method is evaluated from the list of top- N recommendations for each user. In order to evaluate, we use hit rate (HR) and average reciprocal hit rating (ARHR) as evaluation criteria.

- **Hit-rate:** For the experimental set, the hit criteria indicate the number of test items in the recommended items for a user and hit rate specifies the number of hits per user in the entire system. So, for the top- N recommendation, it is:

$$HR@N = \frac{\text{numberofhits}}{\text{numberofusers}}. \quad (11)$$

- **Average reciprocal hit rate:** Now, to improve the recommendation quality, we always want to see the position of the test items in the recommended list. It is preferable to receive the test items at the beginning or near the beginning of the recommended list. Now, for the total number of hits, h , we show the position in the recommended list as $p_1, p_2, \dots, p_i, \dots, p_h$, defining $1 \leq p_i \leq N$ and ARHR as follows

$$ARHR@N = \frac{1}{\text{Numberofusers}} \sum_{i=1}^{i=h} \frac{1}{p_i}. \quad (12)$$

We want to achieve the maximum ARHR, and at best it is possible that all the test items occur in the first position, leading to $ARHR = HR$. The minimum value of $ARHR@N$ is equal to $HR@N/N$, where all test items are placed in the bottom of the list. In addition to these two criteria, we also measure the number of cold-start items in the recommended list. If there are ci_t of cold-start items in the recommended list for user u_t , we measure the preference of cold-start items (CSIP) as follows:

$$CSIP = \frac{\sum_{t=1}^{n_1} ci_t}{\text{numberofusers}}. \quad (13)$$

We want to maximize this CSIP value and its maximum value can be N . However, having the maximum value of CSIP is not preferred, as it seems to recommend only cold-start items, which lose the co-operation and personalization feature of the recommending system.

5.3 Results and discussion

We show the results in three parts, all-user results, cold-start user result and results for cold-start item preference.

5.3.1 All-user results

As can be seen in Table 3, in all datasets, such as the Delicious dataset, the proposed method performs better than all others in terms of hit rate and Average Reciprocal Hit Rate. In the experiments, it is observed that performance improvement for low values of N is higher than other methods which is our main goal. Compared to Banerjee et al. (2021), which

Table 3 Statistics of the additional information

Dataset	#Social-tie	Density	Avg-degree	#Tags	Tag-density	Avg-tag
Last.fm	12717	0.0071	13.44	9749	0.0015	14.89
Delicious	43597	0.0271	48.65	11250	0.0008	26.79
LibraryThing	58385	0.0003	6.22	17228	0.0079	636.37

Table 4 All-user results comparison

Dataset	Algorithm	HR@5	HR@10	HR@15	HR@20	ARHR@5	ARHR@10	ARHR@15	ARHR@20
Last.FM	IBCF	0.0231	0.0534	0.0898	0.1276	0.0114	0.0152	0.0181	0.0197
	IKNN-Tag	0.0136	0.0393	0.0663	0.0919	0.0071	0.0101	0.0123	0.0138
	SVD	0.0466	0.0777	0.0986	0.1107	0.0266	0.0307	0.0323	0.0329
	BPR	0.0568	0.0817	0.1006	0.1195	0.0322	0.0357	0.0371	0.0382
	GBPR	0.0527	0.0750	0.0898	0.1073	0.0275	0.0306	0.0317	0.0327
	SBPR	0.0621	0.0911	0.1093	0.1269	0.0363	0.0401	0.0414	0.0424
	Banerjee et al. (2021)	0.1288	0.1567	0.1760	0.1896	0.0902	0.0940	0.0955	0.0963
	Proposed	0.1305	0.1584	0.1770	0.1901	0.0917	0.0955	0.0965	0.0968
Delicious	IBCF	0.0311	0.0906	0.1870	0.3022	0.0115	0.0191	0.0266	0.0331
	IKNN-Tag	0.0144	0.0401	0.0828	0.1209	0.0057	0.0089	0.0121	0.0143
	SVD	0.0073	0.0150	0.0192	0.0238	0.0035	0.0044	0.0048	0.0051
	BPR	0.0174	0.0234	0.0329	0.0394	0.0089	0.0096	0.0104	0.0107
	GBPR	0.0162	0.0204	0.0323	0.0400	0.0107	0.0113	0.0122	0.0126
	SBPR	0.217	0.2631	0.2976	0.3244	0.1453	0.1517	0.1544	0.1559
	Banerjee et al. (2021)	0.1988	0.2221	0.2393	0.2589	0.1602	0.1631	0.1645	0.1656
	Proposed	0.2003	0.2236	0.2403	0.2594	0.1617	0.1646	0.1655	0.1661
LibraryThing	IBCF	0.0003	0.0004	0.0004	0.0005	0.0003	0.0003	0.0003	0.0003
	IKNN-Tag	0.0057	0.0105	0.0169	0.0228	0.0024	0.0030	0.0035	0.0038
	SVD	0.0055	0.0084	0.0104	0.0123	0.0030	0.0034	0.0036	0.0037
	BPR	0.0057	0.0091	0.0136	0.0163	0.0031	0.0035	0.0038	0.0040
	GBPR	0.0083	0.0134	0.0188	0.0225	0.0040	0.0047	0.0051	0.0053
	SBPR	0.0090	0.0150	0.0193	0.0246	0.0046	0.0054	0.0057	0.0060
	Banerjee et al. (2021)	0.0113	0.0154	0.0196	0.0248	0.0070	0.0075	0.0079	0.0081
	Proposed	0.0128	0.0169	0.0206	0.0253	0.0085	0.0090	0.0089	0.0086

uses tag and social information, the improved performance for our method is evident.

In our proposed method, social information is applied more effectively in recommendation process. Thus this improves HR and ARHR values compared to Banerjee et al. (2021). For low values of N, the impact of social information on performance improvement increases; therefore, methods that make better use of social information overcome other methods. The proposed method has high ARHR values for all datasets and for all N values. It is obvious that HR and ARHR values increase when the value of N increases, and as the length of the list (N) increases, the probability of

hits also increases. The IBCF method achieves better results than IKNN except for the libraryThing dataset, because this dataset has less cuser value and higher tag-density.

For Last.fm dataset, the proposed method has the highest performance in terms of HR and ARHR criteria, because this dataset has a good combination of cuser, average-degree in social network and tag-density. For Delicious dataset, this dataset is heavily imbalanced since it has too many items compared to the number of users, which causes high cuser and low ritem value. Although SBPR performs best due to the high social network density, (Banerjee et al. 2021) cannot overcome SERP because the tag-distribution is not good.

Table 5 Performance comparison for cold-start users

Dataset	Algorithm	HR@5	HR@10	HR@15	HR@20	ARHR@5	ARHR@10	ARHR@15	ARHR@20
Last.FM	IBCF	0.0263	0.0624	0.1026	0.1469	0.0086	0.0130	0.0160	0.0197
	IKNN-Tag	0.0102	0.0424	0.0825	0.1086	0.0039	0.0078	0.0111	0.0126
	SVD	0.0705	0.1046	0.1367	0.1428	0.0419	0.0464	0.0488	0.0491
	BPR	0.0765	0.1106	0.1347	0.1548	0.0372	0.0421	0.0440	0.0452
	GBPR	0.0785	0.1126	0.1347	0.1649	0.0431	0.0476	0.0493	0.0510
	SBPR	0.0845	0.1287	0.1528	0.1689	0.0479	0.0537	0.0556	0.0565
	Banerjee et al. (2021)	0.0986	0.1367	0.1588	0.1749	0.0734	0.0784	0.0802	0.0811
Delicious	Proposed	0.1001	0.1382	0.1598	0.1754	0.0749	0.0799	0.0812	0.0816
	IBCF	0.0112	0.0387	0.0936	0.1376	0.0031	0.0062	0.0105	0.0129
	IKNN-Tag	0.0112	0.0167	0.0661	0.0991	0.0031	0.0038	0.0076	0.0096
	SVD	0.0002	0.0057	0.0057	0.0057	0.0002	0.0010	0.0010	0.0010
	BPR	0.0112	0.0112	0.0167	0.0222	0.0071	0.0071	0.0075	0.0077
	GBPR	0.0167	0.0277	0.0387	0.0551	0.0051	0.0068	0.0077	0.0087
	SBPR	0.2475	0.2914	0.3134	0.3409	0.1886	0.1943	0.1959	0.1975
LibraryThing	Banerjee et al. (2021)	0.3628	0.3903	0.4233	0.4562	0.3053	0.3091	0.3115	0.3133
	Proposed	0.3643	0.3918	0.4243	0.4567	0.3068	0.3106	0.3125	0.3138
	IBCF	0.0002	0.0006	0.0006	0.0006	0.0002	0.0003	0.0003	0.0003
	IKNN-Tag	0.0026	0.0047	0.0079	0.0103	0.0013	0.0015	0.0017	0.0018
	SVD	0.0020	0.0030	0.0035	0.0040	0.0012	0.0013	0.0013	0.0013
	BPR	0.0091	0.0125	0.0171	0.0210	0.0044	0.0048	0.0051	0.0053
	GBPR	0.0095	0.0154	0.0231	0.0273	0.0037	0.0045	0.0051	0.0053
LibraryThing	SBPR	0.0159	0.0231	0.0265	0.0358	0.0095	0.0105	0.0107	0.013
	Banerjee et al. (2021)	0.0205	0.0277	0.0332	0.0366	0.0147	0.0157	0.0161	0.0163
	Proposed	0.022	0.0292	0.0342	0.0371	0.0162	0.0172	0.0171	0.0168

The proposed method uses social network information more effectively, so it achieves better performance than (Banerjee et al. 2021) and SBPR methods in terms of HR and ARHR criteria. For LibraryThing dataset, all methods have low hit rate due to very low rating density and social network density. However, the tag-density and tag-distribution are very good, which makes the tags give better results. As can be seen, in this dataset the IKNN-tag method works better than IBCF, SVD and BPR. The method (Banerjee et al. 2021), which combines item and social tags, works better in terms of HR and ARHR compared with other methods except the proposed method. The proposed method, as can be seen, uses social information more efficient than (Banerjee et al. 2021) and is superior in terms of HR and ARHR.

5.3.2 Cold-start user results

For this evaluation, we consider users who have less than or equal to 5 items in the training set as cold-start users. In all

datasets, the proposed method is superior to other methods in terms of both HR and ARHR, which shows that the use of tags and social information solves the problem of cold-start user. The SBPR method has the best performance compared to the IBCF, IKNN-tag, SVD, BPR and GBPR methods because it considers the communication information of external users. In this dataset, similar to the all-user results section, the HR and ARHR values increase with increasing N. For Delicious dataset, The SBPR method and (Banerjee et al. 2021) have significant growth in performance over the others. Although the SBPR method in the all-user results section has better performance in terms of HR than (Banerjee et al. 2021), the latter method works better than SBPR for cold-start users.

The proposed method achieve higher HR and ARHR values due to effective use of social information than other methods such as (Banerjee et al. 2021) and SBPR for cold-start users. In LibraryThing dataset, for low values of N, the better performance of Banerjee et al. (2021) is evident compared to other methods. In the proposed method, social

information for more users is considered compared to Banerjee et al. (2021), and therefore for low values of N , it is superior to Banerjee et al. (2021) in terms of HR and ARHR, which is desirable. For all datasets, the efficiency of the proposed method is evident in terms of HR and ARHR criteria for all N values (Table 4).

5.3.3 Results for cold-start item preference

In all datasets, the IKNN-Tag has the highest value for CSIP. The IKNN-Tag considers tags to find similarities between items, which makes the cold-start item looks like the most similar existing item. The Delicious dataset has very high values of CSIP, which is due to the features of this dataset that more than 90% of the items are cold-start items. In all methods, GBPR has the lowest CSIP value. Among all datasets, considering the trade-off between performance and CSIP, our proposed method offers a good CSIP value with the highest HR and ARHR values (Table 5).

6 Conclusion

In this paper, we present a new method for recommending N -top items, which uses three different types of information: user-item ranking data, social network between users, and item-related tags. This article also uses structural information and trust among users within the social network and extracts the implicit connections between users and uses them in the item recommendation process. We implement the proposed method with three datasets for recommendations. We compare our results with the ones obtained from some advanced ranking methods and observe that the accuracy of our method for all users and cold-start users improves. Our method can also create more items for cold-start users in the list of recommended items. In all datasets, such as the Delicious dataset, the proposed method performs better than all others in terms of hit rate and Average Reciprocal Hit-Rank. Also in Cold-start user results, for all datasets, the proposed method is superior to other methods in terms of both HR and ARHR, which shows that the use of tags and social information solves the problem of cold-start user. As future work, we can use an ensemble of several node-embedding methods to extract embedding trust features for neighbors to make these features more accurate.

Data availability Datasets related to this article can be found at <https://grouplens.org/datasets/hetrec-2011/> for two publicly available datasets named Last.fm and Delicious and LibraryThing dataset can be found at https://cseweb.ucsd.edu/~jmcauley/datasets.html#social_data.

Declarations

Conflict of interest The authors declare that there are no conflicts of interest regarding the publication of this article.

References

- Agarwal N, Haque E, Liu H, Parsons L (2005) Research paper recommender systems: A subspace clustering approach. In: International Conference on Web-Age Information Management. 475–491.
- Ahmed A, Shervashidze N, Narayanamurthy S, Josifovski V, Smola AJ (2013) Distributed large-scale natural graph factorization. In: Proceedings of the 22nd international conference on World Wide Web, 37–48.
- Backstrom L, Leskovec J (2011) Supervised random walks: predicting and recommending links in social networks. In: Proceedings of the fourth ACM international conference on Web search and data mining. pp. 635–644.
- Banerjee S, Banjare P, Pal B, Jenamani M (2021) A multistep priority-based ranking for top-N recommendation using social and tag information. *J Ambient Intell Humaniz Comput* 12(2):2509–2525
- Belkin M, Niyogi P (2001) Laplacian eigenmaps and spectral techniques for embedding and clustering. *Adv Neural Inf. Process. Syst*, 14
- Bell RM, Koren Y (2007) Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In: Seventh IEEE international conference on data mining (ICDM 2007). 43–52.
- Bengio Y, Courville A, Vincent P (2013) Representation learning: a review and new perspectives. *IEEE Trans Pattern Anal Mach Intell* 35(8):1798–1828
- Bhagat S, Cormode G, Muthukrishnan S (2011) Node classification in social networks. *Social network data analytics*. Springer, Boston, pp 115–148
- Bottou L (1991) Stochastic gradient learning in neural networks. *Proc Neuro-Nimes* 91(8):12
- Burke R (2002) Hybrid recommender systems: survey and experiments. *User Model User-Adapt Interact* 12(4):331–370
- Burke R (2007) Hybrid web recommender system. *Adapt Web*. 377–408
- Covington P, Adams J, Sargin E (2016) Deep neural networks for youtube recommendations. In: Proceedings of the 10th ACM conference on recommender systems. 191–198.
- Cox MAA, Cox TF (2008) Multidimensional scaling. *Handbook of data visualization*. Springer, Berlin, Heidelberg, pp 315–347
- Deshpande M, Karypis G (2004) Item-based top-n recommendation algorithms. *ACM Trans Inf Syst* 22(1):143–177
- Fouss F, Pirotte A, Renders J-M, Saerens M (2007) Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans Knowl Data Eng* 19(3):355–369
- Funk S (2006) Netflix update: try this at home
- Getoor L, Taskar B (2007) *Statistical relational learning*. MIT press, Cambridge
- Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 855–864
- Guo G (2013) Integrating trust and similarity to ameliorate the data sparsity and cold start for recommender systems. In: Proceedings of the 7th ACM conference on Recommender systems. 451–454.
- Keikha MM, Rahgozar M, Asadpour M (2018) Community aware random walk for network embedding. *Knowledge-Based Syst* 148:47–54

- Kondor RI, Lafferty J (2002) Diffusion kernels on graphs and other discrete structures. In: Proceedings of the 19th international conference on machine learning, vol. 2002, 315–322.
- Koren Y (2010) Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Trans Knowl Discov Data* 4(1):1–24
- Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. *J Am Soc Inf Sci Technol* 58(7):1019–1031
- Lin F, Cohen WW (2010) Semi-supervised classification of network data using very few labels. In: 2010 international conference on advances in social networks analysis and mining, 192–199.
- Lin J, Sugiyama K, Kan M-Y, Chua T-S (2013) Addressing cold-start in app recommendation: latent user models constructed from twitter followers. In: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. 283–292.
- Linden G, Smith B, York J (2003) Amazon. com recommendations: item-to-item collaborative filtering. *IEEE Internet Comput* 7(1):76–80
- Mikolov T, Chen K, Corrado G., Dean J (2013) Efficient estimation of word representations in vector space. *arXiv Prepr. arXiv1301.3781*
- Morgan SN, Govender KK (2017) Conceptualizing loyalty in the South African mobile telecommunications industry. *Glob J Manag Bus Res* 4:1273816
- Pan W, Chen L (2013) Gbpr: Group preference based bayesian personalized ranking for one-class collaborative filtering
- Pennington J, Socher R., Manning CD (2014) Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543.
- Perozzi B, Al-Rfou R., Skiena S (2014) Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 701–710.
- Radivojac P et al (2013) A large-scale evaluation of computational protein function prediction. *Nat Methods* 10(3):221–227
- Rendle S, Freudenthaler C, Gantner Z., Schmidt-Thieme L (2012) BPR: Bayesian personalized ranking from implicit feedback. *arXiv Prepr. arXiv1205.2618*
- Roweis ST, Saul LK (2000) Nonlinear dimensionality reduction by locally linear embedding. *Science* 290(5500):2323–2326
- Sarwar BM, Karypis G, Konstan J., Riedl J (2002) Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In: Proceedings of the fifth international conference on computer and information technology, vol. 1, pp. 291–324
- Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T (2008) Collective classification in network data. *AI Mag* 29(3):93
- Son LH (2016) Dealing with the new user cold-start problem in recommender systems: a comparative review. *Inf Syst* 58:87–104
- Tang J, Qu M, Wang M, Zhang M, Yan J., Mei Q (2015) Line: Large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web. 1067–1077.
- Tenenbaum JB, de Silva V, Langford JC (2000) A global geometric framework for nonlinear dimensionality reduction. *Science* 290(5500):2319–2323
- Tsoumakas G, Katakis I (2006) Multi-label classification: an overview dept of informatics. Aristotle Univ, Thessaloniki, Greece
- Vazquez A, Flammini A, Maritan A, Vespignani A (2003) Global protein function prediction from protein-protein interaction networks. *Nat Biotechnol* 21(6):697–700
- Wang X, Jin D, Cao X, Yang L., Zhang W (2016) Semantic community identification in large attribute networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, no. 1.
- Wang X, Cui P, Wang J, Pei J, Zhu W., Yang S (2017) Community preserving network embedding
- Wu Y, Liu X, Xie M, Ester M., Yang Q (2016) CCCF: Improving collaborative filtering via scalable user-item co-clustering. In: Proceedings of the ninth ACM international conference on web search and data mining. 73–82.
- Xue G-R et al. (2005) Scalable collaborative filtering using cluster-based smoothing. In: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval. 114–121.
- Yan S, Xu D, Zhang B, Zhang H-J, Yang Q, Lin S (2006) Graph embedding and extensions: a general framework for dimensionality reduction. *IEEE Trans Pattern Anal Mach Intell* 29(1):40–51
- Yang S-H, Long B, Smola A, Sadagopan N, Zheng Z., Zha H (2011) Like like alike: joint friendship and interest propagation in social networks. In: Proceedings of the 20th international conference on World wide web. 537–546.
- Zhao T, McAuley J., King I (2014) Leveraging social connections to improve personalized ranking for collaborative filtering. In: Proceedings of the 23rd ACM international conference on conference on information and knowledge management, 261–270.
- Zhao WX, Li S, He Y, Chang EY, Wen J-R, Li X (2015) Connecting social media to e-commerce: cold-start product recommendation using microblogging information. *IEEE Trans Knowl Data Eng* 28(5):1147–1159

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.