Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 18(4): 4264–4292.
DOI: 10.3934/mbe.2021214
Received: 22 January 2021
Accepted: 27 April 2021
Published: 17 May 2021

Research article

Friend closeness based user matching cross social networks

Tinghuai Ma[1,*], Lei Guo[1], Xin Wang[2], Yurong Qian[3], Yuan Tian[4] and Najla Al-Nabhan[5]

[1] Nanjing University of information science Technology, Nanjing 210044, China
[2] Huafeng Meteorological Media Group, Beijing 100080, China
[3] Xinjiang University, Urumqi 830008, China
[4] Nanjing Institute of Technology, Jiangsu, Nanjing 211167, China
[5] Department Computer Science, KingSaud University, Riyadh 11362, Saudi Arabia

* **Correspondence:** Email: thma@nuist.edu.cn.

Abstract: The typical aim of user matching is to detect the same individuals cross different social networks. The existing efforts in this field usually focus on the users' attributes and network embedding, but these methods often ignore the closeness between the users and their friends. To this end, we present a friend closeness based user matching algorithm (FCUM). It is a semi-supervised and end-to-end cross networks user matching algorithm. Attention mechanism is used to quantify the closeness between users and their friends. We considers both individual similarity and their close friends similarity by jointly optimize them in a single objective function. Quantification of close friends improves the generalization ability of the FCUM. Due to the expensive costs of labeling new match users for training FCUM, we also design a bi-directional matching strategy. Experiments on real datasets illustrate that FCUM outperforms other state-of-the-art methods that only consider the individual similarity.

Keywords: user matching; cross networks; friend closeness; network embedding; attention mechanism

1. Introduction

In recent years, with the rapid increasement of people's social needs, the dependence on social networks is continued to grow. In the middle of 2020, the total user scale of various social networks around the world was 4.5 billion, and the scale of social network users in China has reached 0.94 billion[*]. In order to take advantage of different services provided by different social networks, it

[*]http://www.cnnic.net.cn/hlwfzyj/hlwxzbg/hlwtjbg/

becomes a common phenomenon that a user may register accounts on multiple social networks respectively. For example, a user can post his latest tweet, feelings, or related news on Twitter to share with others, or use Instagram to share pictures captured at any time. Social network is a platform for communication among users which may have different kinds of forms. The core of Social network is to connect different users, so that users can communicate conveniently to meet the needs of social interaction. The data in these social networking sites can often be used to solve problems in other areas, such as cross-domain recommendation [1, 2], information diffusion [3], privacy protection [4] and even public security field [5, 6]. In terms of cyber-security, it is significant to have a comprehensive social network identity information of a natural individual by using user matching technology. A holographic profile of the user can be quickly established through user matching technology, which can provide rich information support for case investigation, identity verification and risk warning. For example, the verification is more difficult when investigating a suspicious user of a overseas social network. At this time, police can try to match the suspicious user to the known user information so that reduce the workload of manual checking and get more investigation clues. Therefore, it is necessary to match users across social networks. User matching across social networks is an emerging field that has attracted the attention of more and more experts and scholars.

Current cross-network user matching methods mainly focus on the users' attributes and the network structure:

1) Research based on users' attributes mainly uses the statistical characteristics of users. This type of method requires pre-extracting features from user profile or activities, such as user names, interests, and writing style. The current mainstream method is to match based on common user attributes in each social network, such as user name attributes. The latest research on user names mainly includes the model based on prefix or suffix [7], information redundancies based on display names [8] and the model based on sparsity [9]. However, when users use different usernames in different social networks, the above mentioned methods are usually difficult to achieve high-accuracy user matching. In view of this challenge, there are several solutions as follows. Firstly, using the long-term topic interests, language style personalized words and emoticons to match users [10, 11]. The second method is to combine several features extracted from users' posts, such as geographic location, timestamp and language, to describe their identity [12]. At the same time, applied sociology and psychology theory also can be used to model user behavior patterns to map the identity on social networks [11]. However, the user features in these methods usually need to be specified manually, and a deep understanding of domain knowledge is also required.

2) In the research based on the network structure, the user's social network topology is mainly used, such as the friend relationship between users. As one of the main technical methods in this direction, the network embedding method can encode the network in a continuous low-dimensional vector space, while effectively preserving the network structure, such as using dual learning embedding paradigm to achieve the purpose of improving the connection result [13]. Internal links, external links, and cross-network labeled user pairs can be used to generate a probabilistic graph classifier [14] or provide multiple network embedding spaces for association computing [15]. Some approachs use the structural features of social networks for user matching such as common neighbors [16]. Local and global consistency and Adamic/Adar [17] scores in multiple networks is another important method to calculate and measure neighbor similarity. Meanwhile, it is also an effective method for network embedding which uses paired follower-followee relationships to maintain the proximity of users with

similar relationships [18]. In addition, there are incidence matrix [19], hypergraph [15] or network embedding method [20] to solve this problem. Among them, matrix decomposition often involves the inverse or eigenvector of the matrix, which makes it difficult to apply to large-scale data sets.

The current challenges are mainly as follows: 1) Different social network sites are independent, users register accounts in different social networks for different purposes, and their activities in different social networks and the behavior are also different. It is a challenge to provide a unified framework to solve the heterogeneity between different social networks and realize the matching between users across networks. 2) Insufficient labeled user pairs available for training will result in lower training accuracy, but it is costly and extremely time-consuming to generate a large number of accurate and reliable labeled user pairs across social networks. 3) Users in social networks may only have limited personal data, and it is extremely easy to overfit during training, making it difficult to distinguish them from other users.

In order to address the cross-network user matching problem and the above-mentioned challenges more effectively, in this paper, we propose a new method friend closeness based cross-network user matching (FCUM). This method is mainly aimed at solving insufficient attention to the social group relationship formed naturally between people. It uses the attention mechanism to quantify the closeness between users, and then uses the network by designing a joint learning model structure and embedding characteristics to enhance the accuracy and generalization ability of the model. Our experimental results on real-world network datasets show that compared with the state-of-the-art methods, such as DeepLink [13] and ABNE [21]. FCUM has a maximum improvement of 6.9% on $Hits@1$ and a maximum improvement of 12.3% on $Hits@30$.

The main advantages and contributions of our work are as follows:

1) We propose a new semi-supervised model FCUM by integrating user matching and bi-directional matching strategy. Based on the two components, we can improves the accuracy of the model by converting unlabeled users into labeled users iteratively.

2) We can quantify the closeness between users and their friends by attention mechanism, and then form close friends vectors based on the user's network embedding vectors space. The results demonstrate that FCUM can avoid over-fitting the input data and increasing the generalization ability.

The remaining parts of this paper are organized as follows. We will introduce related work in the Section 2. The definition of the problem will be introduced in detail in Section 3. The framework of FCUM will be explained in Section 4. The effectiveness of FCUM will be tested in Section 5. Finally, we will end this article in Section 6.

2. Related works

In recent years, the problem of matching accounts that belong to the same identity among different social networks has become one of the research hotspots. The existing works can be divided into three categories: 1) matching methods based on personal attribute characteristics; 2) matching methods based on network structure characteristics; 3) matching methods based on both personal attributes and network structure.

2.1. Matching methods based on personal profile

In order to match different user identities among different social networks, it uses the user's personal profile, such as username [11], users spatiotemporal pattern [22], user-generated content [23] and writing style [24]. In order to match personal attribute information which belongs to the same identity, Malhotra et al. [25] applied an automatic classifier and they found that the username is one of the most important attributes in this problem. From the perspective of information redundancy, Reza et al. [11] used user names to match related user identities by modeling the naming process and rules.Carmagnola et al. [26] proposed a method CS-UDD, which can associate the personal profile that may belong to same user retrieved from different social networks. However, the username might be selected and modified randomly, which increases the difficulty of completing the task. Based on user name attributes and behavior patterns, Zhang et al. [27] proposed a classification model to perform user matching. Narayanan et al. [28] expressed the user attribute information as an n-dimensional vector, and then used exact matching, partial and fuzzy matching to generate the similarity of these vectors to realize the result. Writing style recognition is another promising way to solve the cahllenge of locating users with multiple accounts while revealing various disguise behaviors. Arvind et al. [24] also used the writing style of user-generated text, such as the grammatical structure and frequency of letters to identify users. But this method can lead to overfitting, especially for short texts like Twitter, because such short texts may involve too many attributes.

In addition, membership information of their community and specific behaviors [29] can also be used to identify and perform matching tasks. Peled et al. [30] extracted three features from user-generated content, which includes the timestamp of posting, the location of posting, and writing style. In this research, it can be found that geographic location is the most obvious feature of matching user identities. Based on spatial distance and textual similarity, Belesiotis et al. matched users in a large dataset by individual posts matching [22]. Li et al. [31] presented a supervised machine learning method to extract the temporal and spatial features, and match user accounts by measuring temporal and spatial similarity . Chaozhou et al. [32] proposed a method which includes a matching function to minimize the surface movement distance which called EMD between users in different networks, and proposed a user alignment model named UUILgan based on generative adversarial networks and a user alignment model named UUILomt based on matrix transformation. Jiang et al. [33] proposed a semi-supervised transfer learning method to predict the behavior of cross-social networks through sparse and overlapping crowds. In addition, in order to detect multi-account users across social networks, Luo et al. [34] proposed a single-class classification method.

Almost all methods based on personal attributes focus on writing style analysis or user behavior inference. However, these methods might not solve the problems of lack of the profiles, such as personal attributes and high time complexity, especially, when facing large-scale social networks.

2.2. Matching methods based on network structure

For the way based on the network structure, it must be pointed out that it is a promising method in solving the challenge of user matching which only needs network structure information to align the network based on labeled users. The existing network-based method embeds the node structure from the local context of the node in order to connect identities across social networks, where the local context of the node can be generated by retaining first-order or second-order neighbors.

Community-based features seem to be a natural choice for user matching problems. Shouling et al. [35] demonstrated the feasibility which uses network structural features in order to match different user identities across different social networks. In this method, they combined the greedy strategy and network structure characteristics to match user identities, and treated the iterative results as new labeled user pairs for further research. Koutra et al. [36] introduced the problem of aligning bipartite graphs and proposed a solution based on gradient descent. In addition, in order to reduce the time complexity of the user matching algorithm in large-scale social networks, Zhou et al. [37] adopted a neighborhood-based network structure feature to measure community similarity by calculating the Adamic/Adar score. Zhang et al. [14] propose a method named CLF to matches user identities by passing the matching information formed by labeled users in the source network to the target network. Korula et al. [36] designed a simple, partial and efficient algorithm that uses network structure information to match user identities. At the same time, they provided theoretical proofs for the performance of the algorithm.

Network embedding is another solution to solve matching problem. Man et al. [20] used network embedding technology to capture the observed underlying structural laws of anchor links, and further learned the cross-network matching used to predict anchor links. Zhou et al. [13] designed DeepLink which is an a semi-supervised algorithm based on deep reinforcement learning and end-to-end. Liu et al. [18] proposed IONE, which also embeds two social networks in a public space to capture the social connections of users. Zhou et al. [38] also proposed an unsupervised scheme FRUI-P, a user recognition algorithm based on friend relations without prior knowledge. Derr et al. [39] uses graph convolutional neural network for signed link prediction. Heimann et al. [40] designed the REGAL framework, which implements graph alignment for network representation learning based on the cross-network matrix decomposition method. Zhao et al. [19] designed a dimensionality reduction algorithm based on the hypergraph to learn the common continuous vector of each user. Sun et al. [41] proposed a self-service algorithm to obtain potential anchor nodes for the next calculation in each iteration.

Compared to the above method, existing methods based on network structure only use several labeled users for supervised training, and might not fully mine non-label user profile, which may leads to insufficient training and low matching efficiency. In addition, many methods such as IONE, only use labeled users to embed and align non-label users [18]. However, their labeled users may deviate after training.

2.3. Methods based on both profile and network structure

In order to improve the performance of cross-social network user matching, user node information and network structure information can be used in combination in this situation. For example, Bartunov et al. [42] proposed a new method, which combines distance-based personal attribute characteristics and neighborhood-based network structure characteristics for joint modeling. Peled et al. [30] extracted 27 attributes, which includes personal profile and network structure. Using user-generated profile information, such as published articles or conversations, as the basis for matching is also a method used to improve matching accuracy. For example, Ma et al. proposed a novel combinatorial term weighting scheme CmTLB [43] based on the term weighting scheme, which combined with the application of sentiment analysis [44]. It can increase the diversity of user information used for matching. In particular, training samples can be generated using sentiment-based session generation techniques [45] to expand the number of training samples to improve the efficiency of the training model. Also, graph-based community detection techniques can be used to optimize the

matching process [46] to adapt to changes in data volume [47]. In addition, Ma et al. propose a community detection technique [48] using most influential nodes to detect community structure, which identifies the best nodes with high influence and is better than other centralized methods. Based on these large numbers of attributes, supervised learning techniques are used to achieve user matching. Tan et al. [15] used hypergraphs to model high-order link relationships, projected the manifolds of two social networks into a common embedding vector space, and combined distance-based personal attribute features to obtain better performance. Kong et al. [16] extracted style-based content features and neighborhood-based network structure features. They designed a supervised aggregation algorithm and a maximum weight matching scheme to rank all potential user identities. The purpose of PCT is to combine personal attributes and network structural characteristics to simultaneously infer the potential corresponding connections of multiple shared entities in the network [49]. In addition, Nie et al. [50] proposed the dynamic core interest matching method named DCIM, which comprehensively considers the user's network structure and user-generated content to measure the similarity of user pairs. Zhong et al. [51] proposed a progressive supervised alignment model CoLink, which combines an attribute-based model and a relationship-based model for joint training. Through these two mutually reinforcing models, CoLink can iteratively align users with a small number of labeled user pairs.

Compared with these works, our method can embed the global network structure, automatically capture the potential meaning of the network structure, and obtain higher matching accuracy. From another perspective, the attention mechanism is used to quantify the closeness between users and friends, which increases the generalization ability of the model. Finally, we designed a semi-supervised model and a bi-directional matching strategy to deal with the problem of insufficient labeled user data.

2.4. Graph embedding

We introduce graph embedding below since they are an important concept in our work.

Graph embedding is a bridge between the original network data and network application tasks, also known as network embedding, network representation learning. It aims to express the nodes in the network into low dimensional vector. So that the vector can have the ability of representation and reasoning in vector space, and can be applied to the vector to network applications. The structure of the network is often complex and diverse. Graph embedding usually mines the neighborhood structure of the network, the high-order similarity of nodes and the community structure of the network. Graph embedding based on network structure aims to make learning nodes represent and retain network structure information. The representation of network can complete many kinds of network analysis tasks, such as node classification, link prediction, important node discovery and so on.

Most of the early graph embedding algorithms are based on spectral clustering. Representative works include PCA [52], Locally Linear Embedding [53, 54], Laplacian Eigenmaps [55, 56], Directed Graph Embedding [57], etc. With the development of word embedding technology word2vec [58–60] in NLP, DeepWalk [61] and Node2vec [62] regard nodes as words and generate node sequences by random walk as sentence. Then, neural language models such as Skip-gram [63] can be applied to these node sequences to obtain graph embedding. HARP [64] improves the solution and avoids local optimization by better weight initialization. It is combined with the method based on random walk to

get better optimization function solution. In order to solve the sparsity of adjacency matrix, LINE [65] considers both first-order proximity and second-order proximity of network nodes. Grarep [66] preserves the k order proximity through a special relation matrix. AROPE [67] learns the network high-order proximity based on SVD. Hope [68] uses JDGSVD algorithm to reduce the dimension of the asymmetric relation matrix of the network to learn the low dimensional representation of nodes. NetHiex [69] cooperatively trains the node representation and class representation, and the node representation is composed of multiple class representations with different granularity.

Currently, more and more methods use deep neural network for graph embedding. SDNE [70] uses deep auto-encoder to preserve the neighbor structure of the node. DNGR [71] combines random walk and deep auto-encoder. The model consists of three components, including random surfing, calculation of PPMI matrix and feature reduction by SDAE. DVNE [72] learned a Gaussian distribution as the representation of nodes in the Wasserstein space, which can preserves the structure information of the network and the uncertainty of the formation and evolution of modeling nodes. GAT [73] applied the self-attention mechanism to graph to learn the representation of network nodes. The node can take into account the characteristics of all other neighbor nodes when updating its own characteristics. GAT does not need to know the structure of the entire network in advance, and can implicitly assign different weights to nodes in the same neighborhood, which is conducive to the interpretability of the model. GraphSAGE [74] is a graph embedding method based on neighbor feature aggregation, which updates the current node's features by gathering the sampled neighbor features. GraphSAGE can quickly generate node representations for new nodes without additional training. VGAE [75] uses GCN encoder and inner product decoder. The input is adjacency matrices, and they rely on GCN to learn high-order dependencies between nodes. Graph2vec [76] regards a graph as a document, and the rooted subgraphs around all nodes in the graph as words. In other words, the way that rooted subgraphs form a graph is the same as the way words form sentences or paragraphs. RDF2Vec [77] uses a language modeling method that extracts unsupervised features from word sequences and applies them to RDF graphs. It uses the local information in the subgraph structure, Weisfeiler-Lehman Subtree RDF and graph walk to generate sequences, and represents the entities in the RDF graph as vectors. GraphGAN [78] proposed graph softmax, which uses GAN to update the expression of network node vectors.

3. Preliminary background

In this section, we introduce the key terms and descriptions in our proposed approach, and present a few formal definition.

3.1. Key terms and descriptions

Key terms and descriptions used in this paper are listed in Table 1.

3.2. Problem definition

Definition 1 (Social network structure diagram). The definition $G^{X/Y} = \left(V^{X/Y}, E^{X/Y}, F^{X/Y}, W^{X/Y}\right)$ is a undirected graph to indicate the social network G^X or G^Y, where $V^{X/Y} = \left\{v_i^{X/Y}\right\}$ is a set of vertices, each vertex $u_i^{X/Y}$ represents a user in the specified social network, $E^{X/Y} = \left\{e_{ij}^{X/Y}\right\}$ is a collection of edges,

Table 1. Key terms and descriptions.

Terms	Description
$v_i^{X/Y}$	User i in network G^X or G^Y
$e_{ij}^{X/Y}$	The edge from user i to user j in the network G^X or G^Y
$w_{ij}^{X/Y}$	The weight of the edge from user i to user j in the network G^X or G^Y
$\vec{v}_i^{X/Y}$	The network embedding vector of user i in network G^X or G^Y
$f_i^{X/Y}$	The collection of close friends of user i in the network G^X or G^Y
$\vec{f}_i^{X/Y}$	Embedding vector of close friends of user i in network G^X or G^Y
$N(v_i^{X/Y})$	User's circle of friends in network G^X or G^Y
L	The collection of labeled nodes in the network G^X or G^Y
$U^{X/Y}$	The collection of non-label nodes in the network G^X or G^Y

each edge $e_{ij}^{X/Y}$ is used to connect two associated vertices $v_i^{X/Y}$ and $v_j^{X/Y}$. Each $f_i^{X/Y}$ in $F^{X/Y} = \left\{f_i^{X/Y}\right\}$ is used to represent the set of close friends of the specified user $v_i^{X/Y}$. $W^{X/Y} = \left\{w_{ij}^{X/Y}\right\}$ is the edge weight set corresponding to the edge set, each edge weight $w_{ij}^{X/Y}$ represents the closeness of the edge connection, if the social network only has ordinary friend relations, then all $w_{ij}^{X/Y} = 1$.

Definition 2 (Network embedding model). Give all users $v_i^{X/Y}(i = 1, 2, \ldots, n)$ and all edges $e_{ij}^{X/Y}(i = 1, 2, \ldots, n; j = 1, 2, \ldots, n; i \neq j)$ in social networks G^X and G^Y respectively, and then network embedding model is used to learn a vector $\vec{v}_i^{X/Y} \in \mathbb{R}^d$ to represent each user, where $v_i^{X/Y} \in G^{X/Y}$, d is the dimension of the vector space and $d \ll \left|V^{X/Y}\right|$. The $\vec{v}_i^{X/Y}$ is the network embedding vector representation of user $v_i^{X/Y} \in G^{X/Y}$.

Definition 3 (User match). Given two different networks $G^{X/Y} = (V^{X/Y}, E^{X/Y}, F^{X/Y}, W^{X/Y})$, the goal of cross-network user matching is to determine whether any $v_i^X \in G^X$ and $v_m^Y \in G^Y$ belong to the same real user, The result of user matching which is denoted as $\psi\left(v_i^X, v_m^Y\right)$ is calculated by

$$\psi(v_i^X, v_m^Y) = \left\{ \begin{array}{l} 1, v_i^X = v_m^Y \\ 0, v_i^X \neq v_m^Y \end{array} \right\} \tag{1}$$

Definition 4 (Bi-directional matching). For each $v_i^{X/Y} \in G^{X/Y}$, and the function $\Phi\left(\vec{v}_i^X\right) \mapsto \vec{v}_j^{\,Y}$ is used to map the node embedding vector in the network G^X to the corresponding node embedding vector in the network G^Y, which indicates that the two node embedding vectors belong to the same real users. Vice versa, the inverse matching function $\Phi^{-1}\left(\vec{v}_j^{\,Y}\right) \mapsto \vec{v}_i^X$ can also be defined.

4. Model framework

In this section, we will introduce FCUM model in detail. The main steps are described as following. Firstly, in order to extend the friend relationship, we take the labeled user pair set L as the basis. Secondly, random walk strategy and Skip-gram model are adopted to sample user nodes and generate user vector space respectively based on extended network. To calculate the friend closeness, we apply the attention mechanism to the extended network. After that, the close friend vector space can be

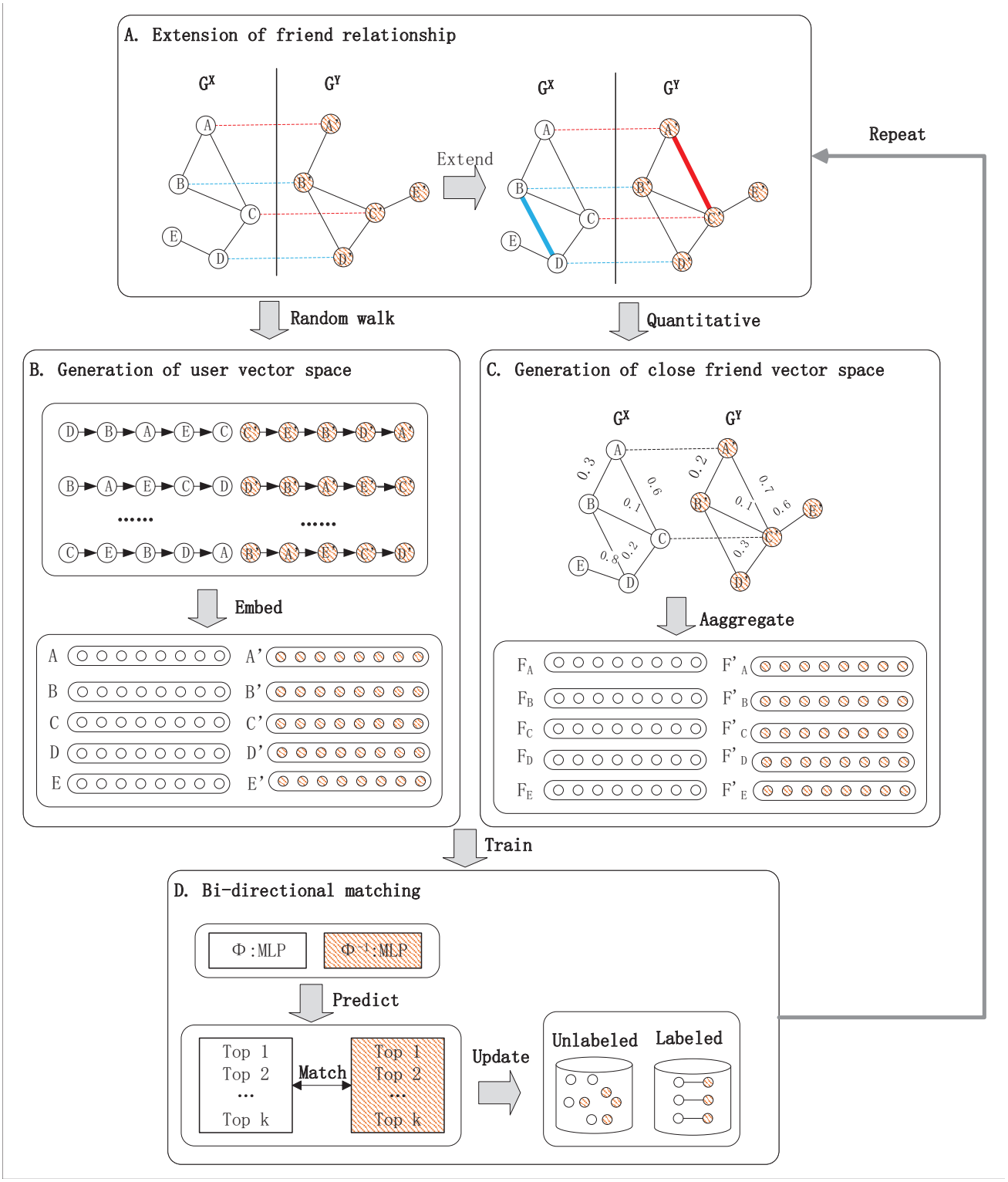


Figure 1. Framework of FCUM.

generated by using user vector space and friend closeness. Then bi-directional matching function is designed to calculate the vectors cosine similarity. Based on the above, the newly matched user node pairs are added to the labeled node pair set L, and remove the corresponding user nodes from the unlabel node pair set U at the same time. At last, the accuracy of FCUM can be gradually improved by iteration with the steps described above.

As shown in Figure 1, FCUM is mainly composed of four main parts: (A) Extension of friend relationship, (B) Generation of user vector space, (C) Generation of close friend vector space, (D) Bi-directional matching. Each part will be described in detail in the sequel.

4.1. Extension of friend relationship

The data we retrieved through web crawlers might be incomplete, and part of the relationship between users in the original data may be lost. Therefore, we use the labeled user pair set L to extend the original edge set $E^{X/Y}$ in the social network before network embedding.

We assume that in the source network G^X, if there is a connected edge between a pair of users, they should also have a corresponding edge in the target network G^Y [79]. According to this assumption, if there is a pair of label nodes $\left(v_i^X, v_m^Y\right) \in L$, v_i^X and v_m^Y are matched cross-network user pairs, and v_i^X and v_j^X in source network G^X has a friend relationship, and at the same time $\left(v_j^X, v_n^Y\right) \in L$ is also a pair of label nodes, v_j^X and v_n^Y are also matched cross-network user pairs, but v_m^Y and v_n^Y in the target network in G^Y doesn't have a friend relationship. At this time, we can reconstruct the edges of v_m^Y and v_n^Y and connect them with edges which is shown in part A of Figure 1. The left network is the network before extension, and the right one is the extended network. The nodes connected by dashed lines represent matched label nodes, and the solid lines being supplemented after extension (B, D) and (A', C') are the missing edges. Through the existing source network G^X, target network G^Y and the label node pair set L, the extended source network G^X and target network G^Y of the friend relationship are calculated by

$$\left\{\begin{array}{l} \widetilde{V}^X = V^X \\ \widetilde{E}^X = E^X \bigcup \left\{ \left(v_i^X, v_j^X\right) : \left\{ \begin{array}{l} \left(v_i^X, v_m^Y\right) \in L \\ \left(v_m^Y, v_n^Y\right) \in E^Y \\ \left(v_j^X, v_n^Y\right) \in L \end{array} \right\} \right\} \\ \widetilde{V}^Y = V^Y \\ \widetilde{E}^Y = E^Y \bigcup \left\{ \left(v_m^Y, v_n^Y\right) : \left\{ \begin{array}{l} \left(v_i^X, v_m^Y\right) \in L \\ \left(v_i^X, v_j^X\right) \in E^X \\ \left(v_j^X, v_n^Y\right) \in L \end{array} \right\} \right\} \end{array}\right. \tag{2}$$

where $\tilde{V}^X = V^X$ means that the extended network contains all the user nodes in the original network G^X.

4.2. Generation of user vector space

Maintaining the user's network structure characteristics as much as possible in the user vector space is essential to improve the accuracy of the final result. Generally speaking, two user nodes with strong correlation, such as having more common friends, should be closer to each other in the user vector space. From the part B of Figure 1, we can observe the two components of the user vector

space generation, includes sampling the network structure and generating user vectors.The graph embedding methods used in this paper are similar to Graph2Vec [76] and Rdf2Vec [77]. The methods mentioned above start with generating sequences related to nodes, and then input them into Skip-gram model for training. But we regard each node in the graph as a word, while Graph2Vec regards the rooted subgraphs around all nodes in the graph as words. And we use random walks for undirected graphs, while Rdf2Vec is mainly for directed labeled RDF graphs. Random walks can not only capture basic network information, but also adapt to the scene of subtle adjustment changes in the network structure without recalculating everything [61]. We can also use this feature to solve the time-consuming problem of generating social sequences in large networks by assigning several threads that work at the same time to perform random walks in parallel. Alibaba also took DeepWalk as the prototype model and introduced side information, and further optimized the model for different side information to form the final solution EGES [80].

The first is to sample the network structure. In order to generate high-quality network embedding, while taking into account the large-scale of social networks and the continuous subtle changes of subsequent user nodes, we adopted random walk strategy [61] to sample user nodes. Starting from any user node $v_i^{X/Y}$, the friend nodes of the currently visited node are selected uniformly and unbiased randomly at each step. Repeat the operation until the number of the visited node reaches the specified length l. For each user node $v_i^{X/Y}$, the above steps will be repeated a certain times t to get multiple social sequences for the user.

The second is to generate the user vector space. We can regard the social sequence generated by sampling the network structure as short sentences of a specific length, and the nodes in the sequence can be regarded as words in a special language. Then we can use the derived Skip-gram model to generate the user vector space. The Skip-gram model was originally used to predict the context of a word by maximizing the average log probability in the field of word representation [58]. This model can generate the embedding vector of the node as an extra-product when updating the weight matrix.

Given a user sequence $\left(v_1^{X/Y}, v_2^{X/Y} \dots v_n^{X/Y}\right) \in G^{X/Y}$, the goal of the Skip-gram model is to maximize the co-occurrence probability of a context node with a certain node as the center node and a sliding window size of c, that is, to maximize the following logarithmic probability, which is calculated by

$$\max \frac{1}{n} \sum_{i=1}^{n} \sum_{j=-c, j\neq 0}^{c} \log p\left(v_{i+j}^{X/Y} \mid v_i^{X/Y}\right) \tag{3}$$

where c is the size of the sliding window, increasing the c of the training context can get a more accurate embedding vector, but it will inevitably consume more training time, so we need to choose an appropriate sliding window size. And $p\left(v_{i+j}^{X/Y} \mid v_i^{X/Y}\right)$ is calculated by the softmax function, which means that the j-th hop neighbor of the given user $v_i^{X/Y}$ appears and the probability is calculated by

$$p\left(v_{i+j}^{X/Y} \mid v_i^{X/Y}\right) = \frac{\exp\left(\left(\vec{v}_{i+j}^{X/Y}\right)^T \cdot \vec{v}_i^{X/Y}\right)}{\sum_{t=1}^{n} \exp\left(\left(\vec{v}_t^{X/Y}\right)^T \cdot \vec{v}_i^{X/Y}\right)} \tag{4}$$

where $n = \left|V^{X/Y}\right|$ is the number of all nodes in $G^{X/Y}$, and $\vec{v}_i^{X/Y}$ is the vector of node $v_i^{X/Y}$, which will be updated in each iteration of training.

The Skip-gram neural network has a large-scale weight matrix. If all the samples are used to adjust these weights, which consumes too much computing resources, and the training process will be very slow. So in order to improve the training efficiency, we use the negative sampling method to sample in a specific noise distribution to avoid all negative samples participating in training. Maximize the objective function is defined as follows [20, 81]

$$\log \sigma\left(\vec{v}_{i+j}^{T} \cdot \vec{v}_i\right) + \sum_{k=1}^{K} E_{v_k \propto \rho_n(v)}\left[\log\left(1 - \sigma\left(\vec{v}_k^{T} \cdot \vec{v}_i\right)\right)\right] \tag{5}$$

The first term represents the probability of edges between nodes v_i and v_{i+j}, and the second term represents the probability of edges between v_i and K negative samples from the noise distribution. According to the research results of Mikolov et al. [58], it is recommended to set $\rho_n(v) \propto d_v^{3/4}$, where d_v is the degree of node v.

4.3. Generation of close friend vector space

At present, some supervised user matching models only use a limited number of label nodes for training, and most of the non-label nodes cannot be used effectively. In order to make full use of non-label nodes, we mine the user friend structure. In some methods, the structural information of the user's friends is often ignored and only focuses on the structural information of the user. If only a small number of label nodes are used for model training under a large-scale network structure, it is likely to lead to overfitting, and thus unable to effectively match users. Therefore, it is very necessary to explore the user's friend structure and use the value of non-label nodes.

The ego-network can reflect a person's social friend environment, but the appearance of different friends to a person is shown in part C of Figure 1, so we need to distinguish between them and choose important friends. For this reason, we use the weight $w_{ij}^{X/Y}$ to measure the closeness between users $v_i^{X/Y}$ and $v_j^{X/Y}$, and assume that the closer the friend is, the greater the influence on a person.

However, most social networks do not distinguish the closeness of users' friends, that is, $w_{ij}^{X/Y} = 1$, which is meaningless. Therefore, we also need to propose a user closeness calculation method in social networks. At present, there are some methods to use graph attention mechanism to distinguish the importance of users [82]. In this article, we use the attention mechanism to recalculate the friend closeness weight $w_{ij}^{X/Y}$ between user $v_i^{X/Y}$ and $v_j^{X/Y}$, which is calculated by

$$w_{ij}^{X/Y} = \frac{\exp\left(\vec{a}^{X/Y} \cdot \left(\vec{v}_i^{X/Y} \circ \vec{v}_j^{X/Y}\right)\right)}{\sum_{v_k \in N^{X/Y}(v_i)} \exp\left(\vec{a}^{X/Y} \cdot \left(\vec{v}_i^{X/Y} \circ \vec{v}_k^{X/Y}\right)\right)} \tag{6}$$

where $\vec{v}_i^{X/Y} \circ \vec{v}_j^{X/Y}$ represents the Hadamard product of two vectors, and $\vec{a}^{X/Y}$ is the vector parameter that needs training and learning. After generating the weight $w_{ij}^{X/Y}$, each user can generate a new attention vector, defined by

$$\vec{v}_i^{*X/Y} = \sum_{v_j \in N^{X/Y}(v_i)} w_{ij}^{X/Y} \cdot \vec{v}_j^{X/Y} \tag{7}$$

The attention vector $\vec{v}_i^{*X/Y}$ can be regarded as the weighted sum of all neighbor vectors of the user. Then the label node set is used as the supervision information, and the target optimization can be

performed by minimizing the following variance loss function to generate the vector parameter $\vec{a}^{X/Y}$, which is calculated by

$$loss_{att} = \frac{1}{2n}\left[\sigma\left(\vec{v}_i^{*X} \cdot \vec{v}_m^{*Y}\right) - y\right]^2 \tag{8}$$

where σ is the sigmoid function, n is the sum of the number of positive and negative samples participating in the training of the attention mechanism, and $y \in (0, 1)$ indicates whether the users v_i and v_m belong to the same real user.

Then, we use Eq (7) to get the weight $w_{ij}^{X/Y}$ that represents the closeness of the users $v_i^{X/Y}$ and $v_j^{X/Y}$. We set an appropriate threshold for it, and then we can filter out the closest friends to the user as the user's close friend set $f_i^{X/Y}$ which is calculated by

$$f_i^{X/Y} = \bigcup_{v_k \in N^{X/Y}(v_i)} \left\{v_m^{X/Y} : w_{ik}^{X/Y} \geq top@\alpha\right\} \tag{9}$$

where α is the friend closeness factor. $top@\alpha$ is the α-th largest weight value of the edge connected to the user node $v_i^{X/Y}$ in the network G^X or G^Y. Only friends greater than or equal to $top@\alpha$ can be used to participate in the aggregation operation.

At last, we use an aggregation operation to generate the friend closeness embedding vector of the user $v_i^{X/Y}$, which is calculated by

$$\vec{f}_i^{X/Y} = \frac{\sum_{v_j \in f_i^{X/Y}} \vec{v}_j^{X/Y}}{\left|f_i^{X/Y}\right|} \tag{10}$$

4.4. Bi-directional matching

As shown in part D of Figure 1, we design the bi-directional matching functions Φ and Φ^{-1}: $\Phi\left(\vec{v}_i^X\right) \mapsto \vec{v}_m^Y$ and $\Phi^{-1}\left(\vec{v}_m^Y\right) \mapsto \vec{v}_i^X$, these functions will be used to determine whether users from different social networks matched or not. In this article, we use two multilayer perceptrons to realize the above matching relationship: $\Phi = MLP\left(\vec{v}_i^X, \theta, b\right)\right)$ and $\Phi^{-1} = MLP^{-1}\left(\vec{v}_m^Y, \theta^{-1}, b^{-1}\right)$, where $(\theta, b, \theta^{-1}, b^{-1})$ are weight parameter and bias term of multilayer perceptrons respectively.

We use a simple four layer MLP as the matching function to map the vectors in the source network G^X to the vectors in the target network G^Y by nonlinear transformation. MLP has simple structure and high efficiency. In addition, if we can achieve good accuracy based on a simple model, it can also verify the contribution of our work on the other hand. For example, the quantification of user's close friend relationship and credible bi-directional matching strategy.

The distance between the prediction result $\Phi_f\left(\vec{v}_i^X\right)$ and the true corresponding vector $\vec{v}_m^Y$ in the target network G^Y should be minimized at the same time, and the loss function is defined by

$$loss_u = \sum_{\left(v_i^X, v_m^Y\right) \in L} \left\|\Phi\left(\vec{v}_i^X\right) - \vec{v}_m^Y\right\|_F \tag{11}$$

and the distance between the prediction result vector $\Phi_f\left(\vec{f}_i^X\right)$ and the true corresponding vector $\vec{f}_j^Y$ in the target network G^Y is calculated by

$$loss_f = \sum_{\left(v_i^X, v_m^Y\right) \in L} \left\| \Phi\left(\vec{f_i^X}\right) - \vec{f_m^Y} \right\|_F \tag{12}$$

where L is the set of labeled nodes, and F represents the $F - norm$.

In the same way, the reverse objective function is calculated by

$$loss_u^{-1} = \sum_{\left(v_i^X, v_m^Y\right) \in L} \left\| \Phi^{-1}\left(\vec{v_m^Y}\right) - \vec{v_i^X} \right\|_F \tag{13}$$

$$loss_f^{-1} = \sum_{\left(v_i^X, v_m^Y\right) \in L} \left\| \Phi^{-1}\left(\vec{f_m^Y}\right) - \vec{f_i^X} \right\|_F \tag{14}$$

The final objective function based on user friend closeness is defined as follows:

$$L = \beta(loss_u + loss_u^{-1}) + (1 - \beta)(loss_f + loss_f^{-1}) + loss_{att} \tag{15}$$

where β is the equilibrium factor which is used to balance the weight of each part of the objective function.

When the user v_i^X in the source network G^X can be mapped to the user v_m^Y in the target network G^Y, and the user v_m^Y in the target network G^Y can be reversely mapped to the user v_i^X in the source network G^X, which means when $\Phi\left(\vec{v_i^X}\right) \mapsto \vec{v_m^Y}$ and $\Phi^{-1}\left(\vec{v_m^Y}\right) \mapsto \vec{v_i^X}$ are satisfied at the same time, we think that the user v_i^X matches user v_m^Y. But in the real situation, it is very difficult to directly map the user v_i^X in the source network G^X to the user v_m^Y in the target network G^Y. For this reason, we need to put the matching value $\Phi\left(\vec{v_i^X}\right)$ and the user vector $\vec{v_m}$ of the target network as parameters into the cosine similarity formula to generate the similarity value, and then rank the similarity from high to low. The vector cosine similarity is calculated by

$$sim\left(\Phi\left(\vec{v_i^X}\right), \vec{v_m^Y}\right) = \frac{\sum_{p=1}^{d} \phi\left(\vec{v_i^X}\right) \times \vec{v_m^Y}}{\sqrt{\sum_{p=1}^{d} \phi\left(\vec{v_i^X}\right)^2} \times \sqrt{\sum_{p=1}^{d} \left(\vec{v_m^Y}\right)^2}} \tag{16}$$

The specific steps of bi-directional matching are shown in Algorithm 1.

From Algorithm 1, we can see that the bi-directional matching strategy has strict conditions. It can ensure the credibility of the newly generated label users. Although there may be a very small number of wrong label users, but our experiment is not obviously affected.

Finally, we add the newly matched user node pair to the labeled node pair set L, and remove the corresponding user nodes from the network $G^{X/Y}$: $L = L \cup \left(v_i^X, v_m^Y\right), U^X = U^X \backslash \left\{v_i^X\right\}, U^Y = U^Y \backslash \left\{v_m^Y\right\}$. In this way, the labeled node pair set can be continuously extended, and the model accuracy can be gradually improved in subsequent iterative training.

4.5. The overall learning algorithm

Based on the models and concepts which are defined above, the overall method is proposed in Algorithm 1.

Algorithm 1: Pseudo-code of bi-directional matching

Input: Mapping function Φ and Φ^{-1}, unlabeled user set U.
Output: All new label user pairs.

for v_i^X *in* U^X **do**
  Forward mapping: $\Phi\left(\vec{v_i}^X\right) \mapsto \vec{v_m}^Y$, generate prediction vector $\vec{v_a}$
  Calculate the similarity between $\vec{v_a}$ and unmatched user vector in target network G^Y by Eq (16)
  Select the user v_m^Y with the highest similarity
  Reverse mapping: $\Phi^{-1}\left(\vec{v_m}^Y\right) \mapsto \vec{v_i}^X$, generate prediction vector $\vec{v_b}$
  Calculate the similarity between $\vec{v_b}$ and unmatched user vector in source network G^X by Eq (16)
  Select the user v_k^X with the highest similarity
  if $v_k^X = v_i^X$ **then**
    Set $\left(v_i^X, v_m^Y\right)$ as a new label user pair
  end
end
return all new label user pairs

Firstly, in the phase of network extension, the labeled node set L can be used to extend the friend relationship of these users in line 3 and 4. In line 6 we use a random walk strategy to sample the extended friend relationship to generate the social sequence of each user. After sampling, the vector embedding technique is used to generate a user vector space based on the social sequence obtained above in line 7, and each embedded vector in the space can represent a user in a network. In order to fully mine the users friend relationship information, in line 8, we also use the graph attention mechanism to quantify the closeness of users friend relationship, which is represented by the number between [0,1]. And the closeness increases as the weight increases. Then we combine the friend closeness with the user vector space, and obtain the user's close friend vector through the aggregation operation in line 9.

After that, we train the user embedding vector and the close friend vector at the same time from G^X to G^Y direction to obtain the matching function Φ. In the same way, reverse training from G^Y to G^X can obtain the reverse matching function Φ^{-1} between line 11 and 17. After obtaining the trained model, we can use the forward matching function Φ and the reverse matching function Φ^{-1} to match the set of unlabeled users in line 19 . Finally, in line 23 the matching results are merged into the labeled user set, and new matching users are removed from the non-label user set.

We will repeat the above steps until the model has no new label users matched.

4.6. User-matching across multiple networks

The core of user matching across multiple networks is to use the chain rule to connect user matching relationships in multiple social networks [13]. For example, if users of $\left(G^X, G^Y\right)$ and $\left(G^X, G^Z\right)$ are known matched through $G^X \to G^Y \to G^Z$, the match result of $\left(G^X, G^Z\right)$ can be obtained from it. It is even possible to use the above-mentioned ideas to indirectly match networks that are difficult to match

Algorithm 2: Pseudo-code of FCUM

Input: Social networks G^X and G^Y.

Output: The prediction result of U.

Initialize $L' = L$
while *($L' \neq \emptyset$)* **do**
  for *(v_i^X, v_m^Y) in L* **do**
    Extend the connecting edges of users v_i^X and v_m^Y
    Generate extended friends of users v_i^X and v_m^Y by Eq (3)
  end
  Using random walk to generate the walking sequence of $v_i^{X/Y}$
  Generate the embedding vector $\vec{v}_i^{X/Y}$ of user $v_i^{X/Y}$ by Eqs (4) and (5)
  Calculate the edge weight $w_{ij}^{X/Y}$ of the edge $e_{ij}^{X/Y}$ by Eq (7)
  Generate the close friend embedding vector $\vec{f}_i^{X/Y}$ of $v_i^{X/Y}$ by Eq (11)
  while *(model not convergence)* **do**
    Using set L to train the model, each batch of data set size is h
    for *(v_i^X, v_m^Y) in $L[n : n + h]$* **do**
      Find $\vec{v}_i^X$, $\vec{f}_i^X$,$\vec{v}_m^Y$, $\vec{f}_m^Y$ corresponding to (v_i^X,v_m^Y)
      Train the model by Eq (16)
    end
  end
  Obtain the forward model Φ and the reverse model Φ^{-1}
  $L' = \emptyset$
  for $v_i^{X/Y}$ *in* $U^{X/Y}$ **do**
    if $\Phi\left(\vec{v}_i^X\right) \mapsto \vec{v_m}^Y$ *and* $\Phi^{-1}\left(\vec{v_m}^Y\right) \mapsto \vec{v}_i^X$ **then**
      $L' = L' \cup \left(v_i^X, v_m^Y\right)$
    end
  end
  $L = L \cup L', U^X = U^X \backslash \left\{v_i^X : v_i^X \in L'\right\}, U^Y = U^Y \backslash \left\{v_m^Y : v_m^Y \in L'\right\}$
end
return the prediction result of U

Table 2. Data set statistics.

Social network	Number of users	Number of edges	Number of labeled node pairs
Twitter	5220	164919	1609
Foursquare	5315	76972	
DBLP-1	2151	6306	2151
DBLP-2	2151	5676	

or cannot be matched, using matched network as "bridges". Assuming that we cannot match the users in $\left(G^X, G^Z\right)$, but the user matching of $\left(G^Y, G^Z\right)$ is known, we can use $\left(G^X, G^Y\right)$ for matching, and in this way, indirect user matching for $\left(G^X, G^Z\right)$ is achieved through $G^X \to G^Y \to G^Z$.

5. Experiments

In this section, a series of comprehensive experiments are executed to evaluate the performance of our proposed method for cross-network user matching. We will use different methods to verify our algorithm.

5.1. Datasets

This experiment uses a real social network dataset collected from Twitter and Foursquare [14, 18]. For privacy reasons, all sensitive personal information in the data set is deleted. The set of labeled user pairs is generated by crawling the Twitter account link on the user's foursquare homepage. We also used the real academic cooperation data set DBLP [83], which was constructed by the DBLP digital library, in which each vertex represents the authors who has published at least one paper on major conferences and journals in the data mining and database communities from January 1990 to February 2011, each side connects two authors who have co-authored at least one paper. Table II lists the statistics of these two data sets. Among them, DBLP-1 and DBLP-2 are the subsets generated after sampling the author relationship in the data set DBLP.

5.2. Methods for comparison

We compared the proposed method with several existing methods based on network embedding, which also only require network structure information.

CRW [14]: CRW (Collective Random Walk)predicts the formation of social links between target network users and the alignment of anchor links between the target network and other external social networks. CRW consists of two stages: (1) Collective link prediction of anchor links and social links; (2) In a random network that is aligned across parts, the link prediction is propagated by methods of collective random walks.

MAG [15]: MAG (Manifold Alignment on traditional Graphs) uses manifold alignment on the graph to map users across the network. MAG constructs a social graph for each network by calculating the paired weights between users and users, and the ranking results of users are obtained through flow pattern alignment.

MAH [15]: MAH (Manifold Alignment on Hypergraph) is a network embedding method which represents a node as a common low-dimensional space. And it can infer the user's correlation by comparing the distance between two cross-network vectors in the embedding space. MAH uses hypergraphs to model higher-order relationships. For target users in one network, MAH ranks all users in another network for the same user based on the inferred probability of users.

INE [18]: INE (Input Network Embedding) is a simplified version of IONE, which only considers the user node vector and the input vector representation to match.

IONE [18]: IONE (Input-Output Network Embedding) is a network embedding and partial network alignment method. In IONE, the following relationship and the followed relationship are used as input and output contexts, and three vector representations are generated together with user nodes. The anchor link is predicted by simultaneously learning the user's follower embedding vector and the followed embedding vector.

ABNE [21]: ABNE (Attention Based Network Embedding) is an attention-based network embedding model. The model includes a masking graph attention mechanism and a structure-preserving embedding algorithm. Through the supervision of anchor pairs, the attention mechanism is used to learn the weights between users. On the basis of learning weights, the algorithm explicitly establishes the contribution probability model between followers and followed.

DeepLink [13]: DeepLink (Deep Learning based Approach) is an algorithm based on deep reinforcement learning and end-to-end, and it is also a semi-supervised user matching learning algorithm. It does not require a lot of feature engineering, uses unbiased random walks to generate embeddings, then uses multi-layer perceptron to map users, and uses the duality of matching between any two networks to study cross-network user matching problems.

5.3. Evaluation mgetrics

In this experiment, we use $Hits@N$ as the evaluation criterion, which is calculated by

$$Hits@N = \frac{|\ CorrUser@N|^X + |\ CorrUser@N|^Y}{|\ UnLabeledUsers\ | * 2} \tag{17}$$

where $|\ CorrUser@N|^{X/Y}$ represents the number of correct matching user pairs among the top N users in the prediction result list in the social network G^X and G^Y. And the total number of unmatched users is represented by $|\ UnLabeledUsers\ |$.

In addition, MAP, AUC and Hit-Precision are used to evaluate the ranking performance of the algorithm and are defined as follows:

$$MAP = (\sum_1^n \frac{1}{ra})/n \tag{18}$$

$$AUC = (\sum_1^n \frac{m+1-ra}{m})/n \tag{19}$$

$$Hit - Precision = (\sum_1^n \frac{m+2-ra}{m+1})/n \tag{20}$$

where $n = |UnLabeledUsers|$, ra is the ranking of positive sample users, and m is the number of negative sample users. The higher the above three ranking performance indicators, the better the ranking performance of the algorithm.

Table 3. Comparison of matching accuracy on Twitter-Foursquare.

	CRW	MAG	MAH	INE	IONE	ABNE	DeepLink	***FCUM***
Hits@1	0	0.0638	0.05	0.1139	0.1899	0.2785	0.3447	**0.4133**
Hits@5	0.0219	0. 1362	0.1219	0.2342	0.3481	0.4525	0.5942	**0.6014**
Hits@9	0.0476	0. 1705	0.1886	0.2880	0.4494	0.5316	0.6609	**0.6911**
Hits@13	0.0538	0. 2081	0.2148	0.3418	0.4968	0.5823	0.6866	**0.7253**
Hits@17	0.07	0. 23	0.22	0.3766	0.5253	0.6203	0.69	**0.7595**
Hits@21	0.0909	0. 2708	0.2513	0.3956	0.5665	0.6392	0.7	**0.7680**
Hits@25	0.11	0. 29	0.285	0.4399	0.5854	0.6614	0.701	**0.8108**
Hits@30	0.1603	0. 3229	0.3	0.4462	0.6044	0.6677	0.7048	**0.8279**

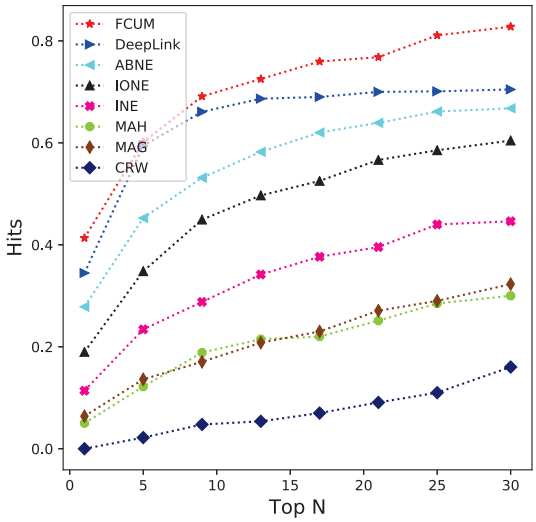


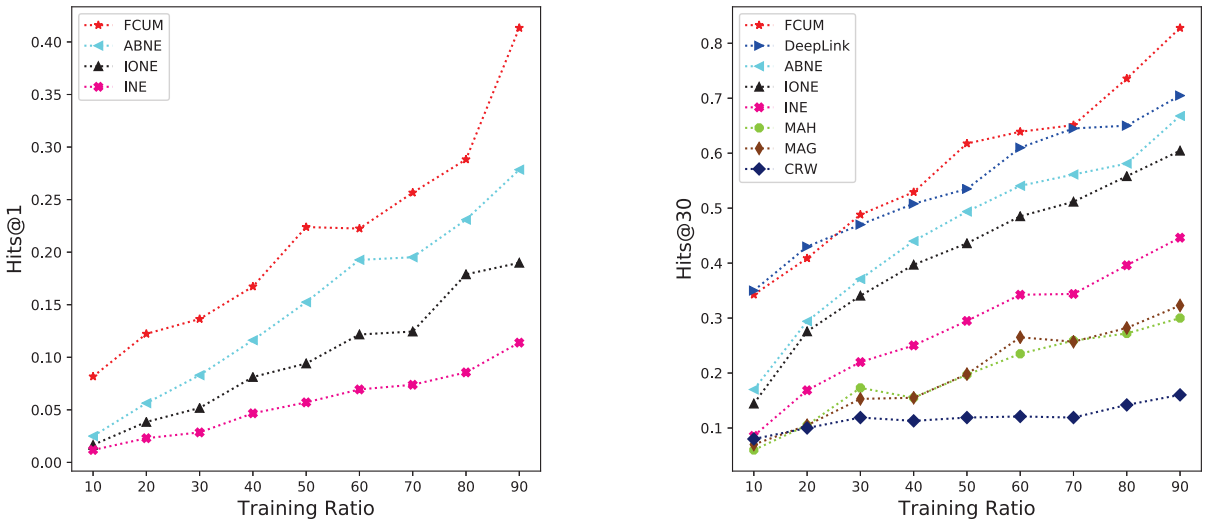
Figure 2. Accuracy on different values of N.

5.4. Evaluation results

First of all, We conduct 10-fold cross-validation by default to test the final model's ability to match tasks across social network users. And then we set the demission to 128, the repeat times of random walks t is 40, the length of random walks l is 80, and the window size of Skip-gram c is 10, the friend closeness factor α to 5 and the equilibrium factor β to 0.2 respectively in our experiment. We will analyze α, β, demission and the important parameters in detail later in this section.

As illustrated in Table III, we examine the performance of various methods on user matching precision on Twitter-Foursquare dataset. Table III reports 8 different N values between 0 and 30, while Figure 2 compares the performance of different methods by varying the value of N.

In Figure 2, we can observe that the CRW model performs the worst compared to several other methods. One of possible reasons is that the adjacency matrix is linked based on the labeled user pair. But in the case of no links between pairs of non-label nodes, the joint matrix will become very sparse, introducing random walks into a "local trap" [21]. Comparing with CRW, the method based on embedding vector shows better results in positioning tasks. For example, MAH tries to learn the



(a) *Hits*@1 accuracy on different training rates

(b) *Hits*@30 accuracy on different training rates

Figure 3. Matching accuracy on different training rates.

embedding vector from the incidence matrix of the hypergraph, and its performance is better than CRW. Our proposed FCUM method is significantly better than all benchmark methods on different *Hits*@*N*, except that the improvement is not obvious than DeepLink on *Hits*@5 where our method is more stable than DeepLink.

In order to evaluate the performance of the model under different training ratios, we set the ratio of training data varies [10%, 90%] on Twitter-Foursquare dataset. The experimental result show in Figure 3.

As can be seen in Figure 3(a), it is obvious that for different training ratios, FCUM outperform all baselines on *Hits*@1. Even if the ratio is set as low as 10 to 20%, the performance enhancement is still significant. As the amount of training data increases, the accuracy is greatly improved. The reason is that the ratio of labeled user pairs used for training will affect the performance of the algorithm. And FCUM can utilize the existing labeled user pairs and the new labeled user pairs generated by subsequent matching process at the same time.

Figure 3(b) shows the performance of FCUM on *Hits*@30. Although the improvement is not obvious than the performance of FCUM on *Hits*@1 in Figure 3(a), but it is still ahead of DeepLink and significantly than other benchmark methods.

As shown in Figure 4, we also studied the influence of the representation dimension of the vector on the experimental accuracy. We can observe that IONE and INE perform better than other models. When the dimension is around 64, it can achieve stable accuracy. When the dimension is greater than 128, the hits of FCUM and ABNE tend to be stable. In fact, 128 is not a big dimension. The dimensions of MAG and MAH need to be around 1000, which close to 10 times of the FCUM model. In general, our FCUM model is stable in smaller dimensions and provides a more accurate embedding for effective similarity calculations. As we all know, the complexity of the learning algorithm highly depends on the dimensionality of the embedding vector. And a small dimension is sufficient for FCUM

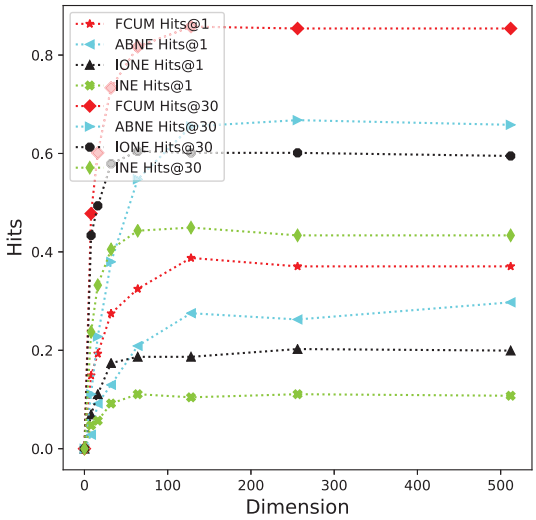


Figure 4. Accuracy on different dimensions.

Table 4. Comparison of ranking performance on Twitter-Foursquare.

	MAP	AUC	Hit-Precision
INE	0.242	0.902	0.902
IONE	0.313	0.948	0.948
ABNE	0.374	0.957	0.957
FCUM	**0.473**	**0.972**	**0.972**

to achieve high accuracy, it is proved that our model is efficient.

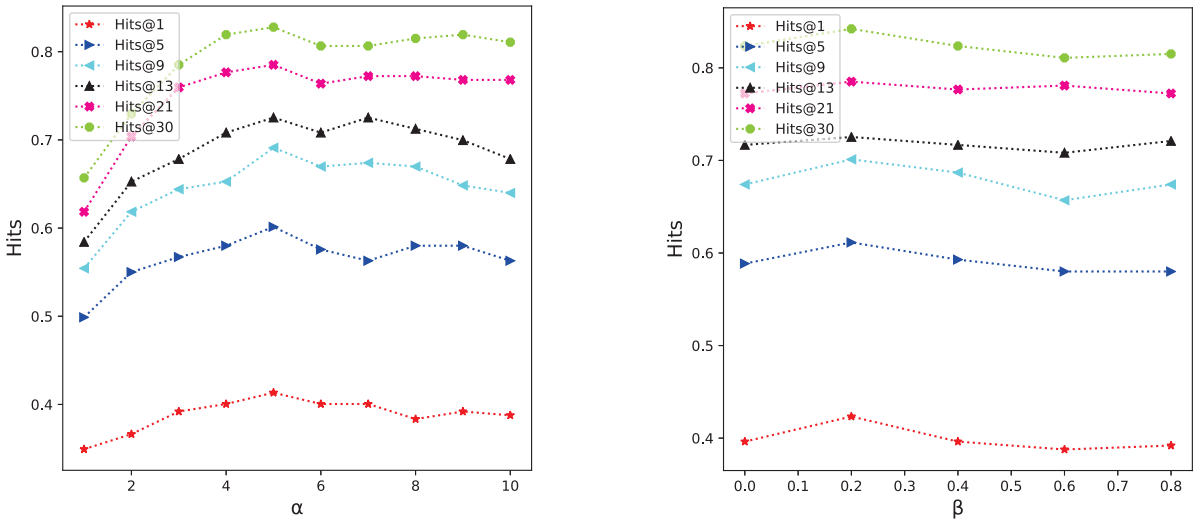
In order to optimize the friend closeness factor α and the equilibrium factor β, the influence of α and β on experimental matching accuracy were analyzed respectively in Figure 5.

We first set β to 0.2 to analyze α. In Figure 5(a), it can be observed that when $\alpha = 5$, the highest accuracy is achieved in different $Hits@N$. This means that aggregating the information of the five closest friends can most effectively improve the matching accuracy. Too many or too few close friends cannot achieve the best results.

Then set α to 5 for β analysis. Figure 5(b) illustrates that by setting β to 0.2, the influence of close friends is increased, and the highest accuracy can be obtained. It also shows the importance of friend closeness used in FCUM.

We also verified the generality and accuracy of our algorithm FCUM on the DBLP dataset in Figure 6. In this experiment, we only used 10% of the labeled nodes to achieve 97.4% accuracy on $Hits@30$, which is 47.2% higher than INE. At $Hits@1$, our algorithm FCUM is also better than ABNE, significantly ahead of other methods. Therefore, our proposed FCUM method has good robustness and effectiveness in cross-social network user matching tasks.

As shown in Table 4, the ranking performance of the algorithm was compared on the Twitter-Foursquare dataset. The proposed algorithm FCUM is also excellent in ranking performance, and has good performance on three indicators. FCUM has a significant lead in the MAP, demonstrating the



(a) The influence of α on matching accuracy

(b) The influence of β on matching accuracy

Figure 5. The influence of hyper parameters on matching accuracy.

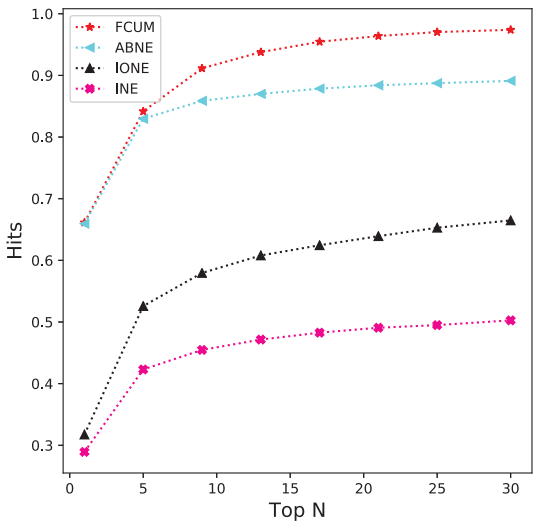
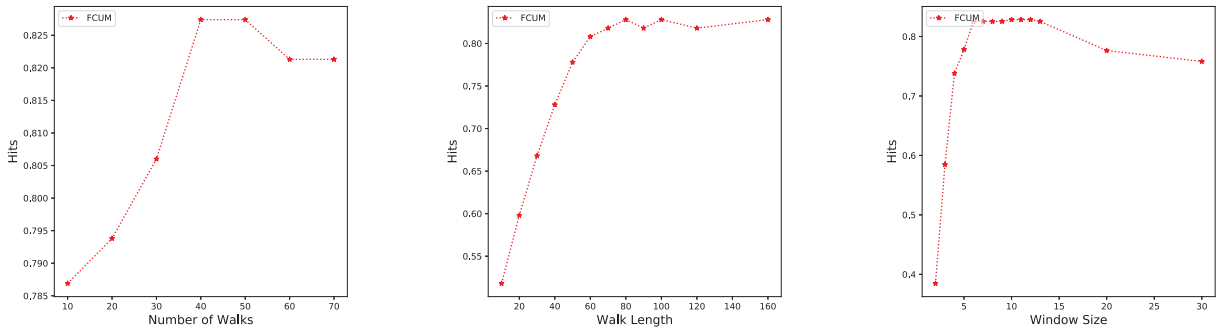


Figure 6. Matching accuracy comparison on DBLP.



(a) Repeat times of random walk t (b) Length of random walk l (c) Window size of Skip-gram c

Figure 7. The influence of important parameters on matching accuracy.

ability to rank absolutely. On the AUC and Hit-Precision that take into account the overall data set size, the relative ranking ability of FCUM is also much better than other comparison methods.

5.5. Parameter analysis

As shown in Figure 7(a), we assign the repeat times of random walks t with values varies [10, 70] to analysis the performance of FCUM. When $t = 10$, the matching accuracy is the lowest. As t gradually increases, the matching accuracy is also greatly improved, reaching the highest value when t is between 40 and 50. Then the matching accuracy decreases, so FCUM can achieve the best performance around c = 40.

As shown in Figure 7(b), we assign the length of random walk l with values varies [10, 160] to analysis the performance of FCUM. When $l = 10$, the matching accuracy is the lowest. As l gradually increases, the matching accuracy also increases, reaching the highest value when $l = 80$. Subsequently, the matching accuracy did not increase significantly with the increase of l. And the larger l, the greater the performance consumption of the machine, so it is appropriate to set $l = 80$.

As shown in Figure 7(c), we assign the window size of Skip-gram c with values varies [2, 30] to analysis the performance of FCUM. When $c = 2$, the matching accuracy is the lowest. As c gradually increases, the matching accuracy increases Significantly, and the matching accuracy can be maintained at the highest point when c around 10. When $c = 20$ and 30, the matching accuracy shows a downward trend, so c can be set to 10.

5.6. Complexity analysis

The time complexity of FCUM is as follows:

$$T = T_{EFR} + T_{GUVS} + T_{GCFVS} + T_{BDM} \tag{21}$$

Where T_{EFR} denotes the time complexity for the extension of friend relationship step. $T_{EFR} = O(|L| \cdot |f_i|)$, where $|L|$ is the number of labeled users, $|f_i|$ is the number of friends of the current user v_i. T_{GUVS} denotes the time complexity for the generation of user vector space step.

$T_{GUVS} = O(k_1 \cdot d \cdot |E|)$, where k_1 is the number of iterations, d is the dimension of user vector, $|E|$ is the number of edges in the network. T_{GCFVS} denotes the time complexity for the generation of close friend vector space step. $T_{GCFVS} = O(k_2 \cdot d \cdot |V| \cdot |f_i|)$, where k_2 is the number of iterations, $|V|$ is the number of social network users. T_{BDM} denotes the time complexity for the bi-directional matching step. $T_{BDM} = O(k_3 \cdot d \cdot |L|) + O(|V|^2)$, where $O(k_3 \cdot d \cdot |L|)$ is the time complexity for MLP training, $O(|V|^2)$ is the time complexity for user matching, k_3 is the number of iterations.

It can be seen that the complexity of T_{EFR} is lower than the other three stages. The numbers of iterations k_1, k_2, k_3 are constant, $|V| \cdot |f_i| = 2|E|$. Therefore, the complexity of T_{GUVS} and T_{GCFVS} is similar. So the complexity of T_{GUVS}, T_{GCFVS} and T_{BDM} depends on $k \cdot d \cdot |f_i|$ and $|V|$. Therefore, in large-scale social networks, T_{BDM} is the dominant complexity. On the contrary, T_{GUVS} and T_{GCFVS} are the dominant complexity.

We used a PC with Windows 10, 2.9-GHz CPU and 8-GB memory. The time cost for the extension of friend relationship step is 3.89 s. The time cost for the generation of user vector space step is 25.73 min. The time cost for the generation of close friend vector space step is 22.41 min. The time cost for the bi-directional matching step is 13.55 min.

6. Conclusions and future works

In this article, we propose a friend closeness based user matching algorithm-FCUM. It is a semi-supervised and end-to-end user matching algorithm. FCUM first extends the social network, and then use the attention mechanism to quantify the closeness between users and friends. After that FCUM generates close friend vectors based on the user embedding vector space. The close friend vectors can be used to avoid overfitting and increase the generalization ability of the model. In addition, we also design a bi-directional matching mechanism to enhance the labeled user pairs set, thereby improving user matching accuracy. Experiments on real-world social network datasets demonstrate that FCUM outperforms various state-of-the-art user matching methods.

The proposed FCUM model's applicability to different social network is a promising dimension. Besides, user matching with multiple features is another interesting direction, such as users' published articles or conversations. But too many features are accompanied by noise, so it is a challenge to match user cross networks. We leave these as future work.

7. Acknowledgements

This work was supported in part by National Science Foundation of China (No. U1736105), and was also supported in part by National Key Research and Development Program of China (2021YFE0104400). The authors extend their appreciation to the Deanship of Scientific Research at King Saud University for funding this work through Research Group no. RG-1441-331.

References

1. C. T. Lu, S. Xie, W. Shao, L. He, S. Y. Philip, Item recommendation for emerging online businesses, in *Ijcai*, (2016), 3797–3803.

2. W. Zhou, W. Han, Personalized recommendation via user preference matching, *Inf. Process. Manage.*, **56** (2019), 955–968.
3. A. Guille, H. Hacid, C. Favre, D. A. Zighed, Information diffusion in online social networks: A survey, *ACM Sigmod Rec.*, **42** (2013), 17–28.
4. I. Nurgaliev, Q. Qu, S. M. H. Bamakan, M. Muzammal, Matching user identities across social networks with limited profile data, *Front. Comput. Sci.*, **14** (2020), 146809.
5. J. Qian, X. Y. Li, C. Zhang, L. Chen, De-anonymizing social networks and inferring private attributes using knowledge graphs, in *IEEE Infocom-the IEEE International Conference on Computer Communications*, IEEE, (2016).
6. Z. Yin, T. Xu, H. Zhu, C. Zhu, E. Chen, H. Xiong, Matching of social events and users: a two-way selection perspective, *World Wide Web*, **23** (2020), 853–871.
7. R. Zafarani, H. Liu, Connecting corresponding identities across communities, in *Proceedings of the International AAAI Conference on Web and Social Media*, (2009), 354–357.
8. Y. Li, Y. Peng, Z. Zhang, H. Yin, Q. Xu, Matching user accounts across social networks based on username and display name, *World Wide Web*, **22** (2019), 1075–1097.
9. D. Perito, C. Castelluccia, M. A. Kaafar, P. Manils, How unique and traceable are usernames?, in *International Symposium on Privacy Enhancing Technologies Symposium*, Springer, (2011), 1–17.
10. S. Liu, S. Wang, F. Zhu, J. Zhang, R. Krishnan, Hydra: Large-scale social identity linkage via heterogeneous behavior modeling, in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, (2014), 51–62.
11. R. Zafarani, H. Liu, Connecting users across social media sites: a behavioral-modeling approach, in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2013), 41–49.
12. O. Goga, H. Lei, S. H. K. Parthasarathi, G. Friedland, R. Sommer, R. Teixeira, Exploiting innocuous activity for correlating users across sites, in *Proceedings of the 22nd International Conference on World Wide Web*, (2013), 447–458.
13. F. Zhou, L. Liu, K. Zhang, G. Trajcevski, J. Wu, T. Zhong, Deeplink: A deep learning approach for user identity linkage, in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, IEEE, (2018), 1313–1321.
14. J. Zhang, S. Y. Philip, Integrated anchor and social link predictions across social networks, in *Twenty-fourth international joint conference on artificial intelligence*, (2015).
15. S. Tan, Z. Guan, D. Cai, X. Qin, J. Bu, C. Chen, Mapping users across networks by manifold alignment on hypergraph, in *Proceedings of the AAAI Conference on Artificial Intelligence*, Citeseer, (2014).
16. X. Kong, J. Zhang, P. S. Yu, Inferring anchor links across multiple heterogeneous social networks, in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, (2013), 179–188.
17. Y. Zhang, J. Tang, Z. Yang, J. Pei, P. S. Yu, Cosnet: Connecting heterogeneous social networks with local and global consistency, in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2015), 1485–1494.

18. L. Liu, W. K. Cheung, X. Li, L. Liao, Aligning users across social networks using network embedding, in *Ijcai*, (2016), 1774–1780.
19. W. Zhao, S. Tan, Z. Guan, B. Zhang, M. Gong, Z. Cao, et al., Learning to map social network users by unified manifold alignment on hypergraph, *IEEE Trans. Neural Networks Learn. Syst.*, **29** (2018), 5834–5846.
20. T. Man, H. Shen, S. Liu, X. Jin, X. Cheng, Predict anchor links across social networks via an embedding approach, in *Ijcai*, (2016), 1823–1829.
21. L. Liu, Y. Zhang, S. Fu, F. Zhong, J. Hu, P. Zhang, Abne: an attention-based network embedding for user alignment across social networks, *IEEE Access*, **7** (2019), 23595–23605.
22. A. Belesiotis, D. Skoutas, C. Efstathiades, V. Kaffes, D. Pfoser, Spatio-textual user matching and clustering based on set similarity joins, *VLDB J.*, **27** (2018), 297–320.
23. C. Riederer, Y. Kim, A. Chaintreau, N. Korula, S. Lattanzi, Linking users across domains with location data: Theory and validation, in *Proceedings of the 25th International Conference on World Wide Web*, (2016), 707–719.
24. A. Narayanan, H. Paskov, N. Z. Gong, J. Bethencourt, E. Stefanov, E. C. R. Shin, et al., On the feasibility of internet-scale author identification, in *2012 IEEE Symposium on Security and Privacy*, IEEE, (2012), 300–314.
25. A. Malhotra, L. Totti, W. Meira, P. Kumaraguru, V. Almeida, Studying user footprints in different online social networks, in *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, IEEE, (2012), 1065–1070.
26. F. Carmagnola, F. Osborne, I. Torre, User data discovery and aggregation: The cs-udd algorithm, *Inf. Sci.*, **270** (2014), 41–72.
27. H. Zhang, M.-Y. Kan, Y. Liu, S. Ma, Online social network profile linkage, in *Asia Information Retrieval Symposium*, Springer, (2014), 197–208.
28. A. Narayanan, V. Shmatikov, De-anonymizing social networks, in *2009 30th IEEE symposium on security and privacy*, IEEE, (2009), 173–187.
29. T. Iofciu, P. Fankhauser, F. Abel, K. Bischoff, Identifying users across social tagging systems, in *Proceedings of the International AAAI Conference on Web and Social Media*, (2011).
30. O. Peled, M. Fire, L. Rokach, Y. Elovici, Matching entities across online social networks, *Neurocomputing*, **210** (2016), 91–106.
31. Y. Li, Z. Zhang, Y. Peng, H. Yin, Q. Xu, Matching user accounts based on user generated content across social networks, *Future Gener. Comput. Syst.*, **83** (2018), 104–115.
32. C. Li, S. Wang, P. S. Yu, L. Zheng, X. Zhang, Z. Li, et al., Distribution distance minimization for unsupervised user identity linkage, in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, (2018), 447–456.
33. M. Jiang, P. Cui, N. J. Yuan, X. Xie, and S. Yang, Little is much: Bridging cross-platform behaviors through overlapped crowds, in *Proceedings of the AAAI Conference on Artificial Intelligence*, (2016), 13–19.
34. X. Luo, F. Zhou, M. Liu, Y. Liu, C. Xiao, Efficient multi-account detection on ugc sites, in *2016 IEEE Symposium on Computers and Communication (ISCC)*, IEEE, (2016), 450–455.

35. S. Ji, W. Li, M. Srivatsa, J. S. He, R. Beyah, Structure based data de-anonymization of social networks and mobility traces, in *International Conference on Information Security*, Springer, (2014), 237–254.
36. N. Korula, S. Lattanzi, An efficient reconciliation algorithm for social networks, preprint, arXiv:1307.1690.
37. X. Zhou, X. Liang, H. Zhang, and Y. Ma, Cross-platform identification of anonymous identical users in multiple social media networks, *IEEE Trans. Knowl. Data Eng.*, **28** (2016), 411–424.
38. X. Zhou, X. Liang, X. Du, J. Zhao, Structure based user identification across social networks, *IEEE Trans. Knowl. Data Eng.*, **30** (2018), 1178–1191.
39. T. Derr, Y. Ma, and J. Tang, Signed graph convolutional networks, in *2018 IEEE International Conference on Data Mining (ICDM)*, IEEE, (2018), 929–934.
40. M. Heimann, H. Shen, T. Safavi, D. Koutra, Regal: Representation learning-based graph alignment, in *Proceedings of the 27th ACM international conference on information and knowledge management*, (2018), 117–126.
41. Z. Sun, W. Hu, Q. Zhang, Y. Qu, Bootstrapping entity alignment with knowledge graph embedding, in *Ijcai*, (2018), 4396–4402.
42. S. Bartunov, A. Korshunov, S.-T. Park, W. Ryu, H. Lee, Joint link-attribute user identity resolution in online social networks, in *Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining, Workshop on Social Network Mining and Analysis. ACM*, (2012).
43. T. Ma, R. Al-Sabri, L. Zhang, B. Marah, N. Al-Nabhan, The impact of weighting schemes and stemming process on topic modeling of arabic long and short texts, in *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, (2020), 1–23.
44. H. Rong, T. Ma, J. Cao, Y. Tian, A. Al-Dhelaan, M. Al-Rodhaan, Deep rolling: A novel emotion prediction model for a multi-participant communication context, *Inf. Sci.*, **488** (2019), 158–180.
45. T. Ma, H. Yang, Q. Tian, Y. Tian, N. Al-Nabhan, A hybrid chinese conversation model based on retrieval and generation, *Future Gener. Comput. Syst.*, **114** (2021), 481–490.
46. T. Ma, W. Shao, Y. Hao, J. Cao, Graph classification based on graph set reconstruction and graph kernel feature reduction, *Neurocomputing*, **296** (2018), 33–45.
47. T. Ma, Y. Zhao, H. Zhou, Y. Tian, A. Al-Dhelaan, M. Al-Rodhaan, Natural disaster topic extraction in sina microblogging based on graph analysis, *Expert Syst. Appl.*, **115** (2019), 346–355.
48. T. Ma, Q. Liu, J. Cao, Y. Tian, A. Al-Dhelaan, M. Al-Rodhaan, Lgiem: Global and local node influence based community detection, *Future Gener. Comput. Syst.*, **105** (2020), 533–546.
49. J. Zhang, P. S. Yu, Pct: partial co-alignment of social networks, in *Proceedings of the 25th International Conference on World Wide Web*, (2016), 749–759.
50. Y. Nie, Y. Jia, S. Li, X. Zhu, A. Li, and B. Zhou, "Identifying users across social networks based on dynamic core interests," *Neurocomputing*, vol. 210, pp. 107–115, 2016.
51. Z. Zhong, Y. Cao, M. Guo, Z. Nie, Colink: An unsupervised framework for user identity linkage, in *Proceedings of the AAAI Conference on Artificial Intelligence*, (2018), 5714–5721.
52. I. Jolliffe, Principal component analysis, *Technometrics*, **45** (2003), 276.

53. S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, *Science*, **290** (2000), 2323–2326.

54. L. K. Saul, S. T. Roweis, An introduction to locally linear embedding, unpublished. Available at: http://www. cs. toronto. edu/˜ roweis/lle/publications. html.

55. M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, in *Nips*, (2001), 585–591.

56. L. Tang, H. Liu, Leveraging social media networks for classification, *Data Min. Knowl. Dis.*, **23** (2011), 447–478.

57. M. Chen, Q. Yang, X. Tang, Directed graph embedding, in *Ijcai*, (2007), 2707–2712.

58. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, preprint, arXiv:1310.4546.

59. T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, preprint, arXiv:1301.3781.

60. T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, S. Khudanpur, Recurrent neural network based language model, in *Eleventh annual conference of the international speech communication association*, (2010).

61. B. Perozzi, R. Al-Rfou, and S. Skiena, Deepwalk: Online learning of social representations, in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, (2014), 701–710.

62. A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, (2016), 855–864.

63. W. Cheng, C. Greaves, and M. Warren, From n-gram to skipgram to concgram, *Int. J. Corpus Linguist.*, **11** (2006), 411–433.

64. H. Chen, B. Perozzi, Y. Hu, S. Skiena, Harp: Hierarchical representation learning for networks, in *Proceedings of the AAAI Conference on Artificial Intelligence*, (2018).

65. J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in *Proceedings of the 24th international conference on world wide web*, (2015), 1067–1077.

66. S. Cao, W. Lu, Q. Xu, Grarep: Learning graph representations with global structural information, in *Proceedings of the 24th ACM international on conference on information and knowledge management*, (2015), 891–900.

67. Z. Zhang, P. Cui, X. Wang, J. Pei, X. Yao, W. Zhu, Arbitrary-order proximity preserved network embedding, in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, (2018), 2778–2786.

68. M. Ou, P. Cui, J. Pei, Z. Zhang, W. Zhu, Asymmetric transitivity preserving graph embedding, in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, (2016), 1105–1114.

69. J. Ma, P. Cui, X. Wang, W. Zhu, Hierarchical taxonomy aware network embedding, in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, (2018), 1920–1929.
70. D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, (2016), 1225–1234.
71. S. Cao, W. Lu, Q. Xu, Deep neural networks for learning graph representations, in *Proceedings of the AAAI Conference on Artificial Intelligence*, (2016).
72. D. Zhu, P. Cui, D. Wang, W. Zhu, Deep variational network embedding in wasserstein space, in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, (2018), 2827–2836.
73. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, preprint, arXiv:1710.10903.
74. W. L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, preprint, arXiv:1706.02216.
75. T. N. Kipf, M. Welling, Variational graph auto-encoders, preprint, arXiv:1611.07308.
76. A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, S. Jaiswal, graph2vec: Learning distributed representations of graphs, preprint, arXiv:1707.05005.
77. P. Ristoski, H. Paulheim, Rdf2vec: Rdf graph embeddings for data mining, in *International Semantic Web Conference*, Springer, (2016), 498–514.
78. H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, et al., Graphgan: Graph representation learning with generative adversarial nets, in *Proceedings of the AAAI conference on artificial intelligence*, (2018).
79. M. Bayati, M. Gerritsen, D. F. Gleich, A. Saberi, Y. Wang, Algorithms for large, sparse network alignment problems, in *2009 Ninth IEEE International Conference on Data Mining*, IEEE, (2009), 705–710.
80. J. Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao, D. L. Lee, Billion-scale commodity embedding for e-commerce recommendation in alibaba, in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, (2018), 839–848.
81. A. Mnih, Y. W. Teh, A fast and simple algorithm for training neural probabilistic language models, preprint, arXiv:1206.6426.
82. L. Sang, M. Xu, S. Qian, X. Wu, Aaane: Attention-based adversarial autoencoder for multi-scale network embedding, in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, (2019), 3–14.
83. A. Prado, M. Plantevit, C. Robardet, J.-F. Boulicaut, Mining graph topological patterns: Finding covariations among vertex descriptors, *IEEE Trans. Knowl. Data Eng.*, **25** (2013), 2090–2104.