MEgo2Vec: Embedding Matched Ego Networks for User Alignment Across Social Networks*

Jing Zhang, Bo Chen, Xianming Wang
Information School, Renmin University of China
{zhang-jing,bochen,wxm}@ruc.edu.cn

Hong Chen*, Cuiping Li, Fengmei Jin
Information School, Renmin University of China
{chong,licuiping,jinfm}@ruc.edu.cn

Guojie Song
MOE Key Laboratory of Machine Perception, PKU
gjsong@pku.edu.cn

Yutao Zhang
Tsinghua University
yt-zhang13@mails.tsinghua.edu.cn

ABSTRACT

Aligning users across multiple heterogeneous social networks is a fundamental issue in many data mining applications. Methods that incorporate user attributes and network structure have received much attention. However, most of them suffer from error propagation or the noise from diverse neighbors in the network. To effectively model the influence from neighbors, we propose a graph neural network to directly represent the ego networks of two users to be aligned into an embedding, based on which we predict the alignment label. Three major mechanisms in the model are designed to unitedly represent different attributes, distinguish different neighbors and capture the structure information of the ego networks respectively.

Systematically, we evaluate the proposed model on a number of academia and social networking datasets with collected alignment labels. Experimental results show that the proposed model achieves significantly better performance than the state-of-the-art comparison methods (+3.12-30.57% in terms of F1 score).

CCS CONCEPTS

• **Information systems** → *Entity resolution*;

KEYWORDS

Social network; Network Integration; User Linkage

ACM Reference Format:
Jing Zhang, Bo Chen, Xianming Wang, Hong Chen*, Cuiping Li, Fengmei Jin, Guojie Song, and Yutao Zhang. 2018. MEgo2Vec: Embedding Matched Ego Networks for User Alignment Across Social Networks. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271705

1 INTRODUCTION

The proliferation of heterogeneous social media platforms, such as Facebook, Twitter, LinkedIn, Instagram, etc., enables us to satisfy our different interests. A study conducted by IPG Media Lab shows that over 50% US adults use more than one social media platform[1]. Aligning the same users across different social networks has recently been attracting considerable attention, as it can provide a comprehensive view of users' profile, and then benefit many applications. For example, linking users of a E-Commerce system and a social media platform can enable product recommendation from the E-Commerce system to the social media platform, and linking users across different social networks can tell us how the information is diffused between them.

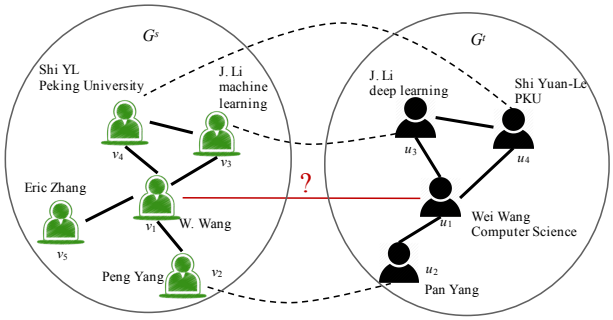


Figure 1: Illustration of the user alignment task. Given two networks G^s and G^t, we aim at aligning user v_1 from G^s and user u_1 from G^t. The lines between users within each network represent the relationships between users. The lines between users across the two networks denote whether two users are aligned or not, where the black dashed lines mean they may be the same person, but the actual labels are unknown, and the red solid line represents the alignment between the target users to be predicted.

One intuitive way to align same users across social networks is to compare the profile attributes such as name, affiliation, description, etc. However, the profile attributes usually cannot provide sufficient evidence. For example, in Figure 1, it is difficult to determine that v_1 and u_1 are the same person only according to their names "W. Wang" and "Wei Wang". Fortunately, the neighbors of the two users in their respective ego networks[2] may provide additional clues. Among the neighbors of v_1 and u_1, "Shi YL" from "Peking University" and 'Shi Yuan-Le" from "PKU" have a high possibility to be a same person, similarly for "J. Li" being interested

*Produces the permission block, and copyright information

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3271705

[1] https://www.ipglab.com/wp-content/uploads/2014/05/A-Network-for-Every-Interest-140-Proof_IPG-Whitepaper.pdf
[2] An ego network consists of a focal node ("ego") and the nodes to whom ego is directly connected to plus the ties between them.

in "machine learning" and "J. Li" being interested in "deep learning". The clues would be more powerful if there are more potential same neighbor pairs. In the Figure, the neighbors of v_1 and u_1 that have a high possibility to be the same person are linked by black dashed lines. In practice, most of the neighbor pairs are unlabeled, which prevents us from directly calculating the metrics like Jaccard's coefficient or Adamic/Adar based on the shared neighbors as that proposed in [7, 8, 26, 30]. One way to leverage the unlabeled information is to estimate the possibility that a user pair can be matched, based on which we can update the matching scores of its neighbor pairs [9, 27, 28]. For example, SiGMa [9] proposed an unsupervised method to iteratively propagate the matching scores from a confidential seed set of user pairs to their neighbor pairs. COSNET [28] proposed a supervised method to infer the marginal probabilities that two users can be matched based on the attributes of the two users and the marginal probabilities of their neighbor pairs. Essentially, to predict the label of a user pair, they both need to infer the labels of its neighbor pairs, and this may cause error propagations during the iterative process.

Despite existing studies on user alignment [7–9, 26–28, 30], the problem still poses several unsolved challenges. First, **how to unitedly model profile similarity?** Traditional methods usually compare different profile attributes by human defined metrics, such as Jaro-Winkler similarity [25], TF-IDF based cosine similarity [7, 20], etc. However, they cannot capture the semantics of different literal strings. Thus, a unified way with little effort of feature engineering to better represent different profile attributes is worth studying. Second, **How to deal with diverse neighbors in different social networks?** The neighbors of a user in different networks can be very diverse due to the various purposes of different social media platforms. For instance, people use Twitter to communicate with friends and obtain information, while they use LinkedIn to find a job. Leveraging all neighbors' information without distinction may contrarily bring in additional noise. Third, **How to incorporate the influence of network topologies?** Different topologies of ego networks may exert different effects. For example, in Figure 1, neighbor pair (v_3, u_3) is linked to another neighbor pair (v_4, u_4), i.e., v_3 is v_4's friend in G^s and u_3 is u_4's friend in G^t. The linkage reduces the possibility that v_3 is wrongly matched to u_3, and v_4 is wrongly matched to u_4, and it further provides strong evidence that v_1 and u_1 are the same person, compared with the situation that (v_3, u_3) and (v_4, u_4) are not linked.

To deal with the above challenges, we formalize the task of user alignment across social networks as a classification problem using the ego networks of two users as input. We propose a graph neural network model—MEgo2Vec to represent the matched ego network between the two users into a low-dimensional real-valued representation, based on which we align the users. The representation includes two parts, one is embedded from the attributes of the target user pairs and their neighbor pairs, the other is embedded from the topologies of the matched ego network. Specifically, first, **to unitedly model the profile attributes**, we adopt a multi-view embedding mechanism to represent the character-view and word-view features for an attribute together. Second, **to tackle the issue of diverse neighbors in different networks**, we adopt attention mechanisms to convolve neighbors' attributes, so as to distinguish their influence. Three attentions are proposed to respectively consider the individual characteristics of each user in a pair, the similarity between two users in a pair, and the relationship of a neighbor pair with the user pair to be predicted. Third, **to capture the influence of network topologies**, we adopt a CNN network to convolve the adjacency matrix of the matched ego network.

The main contributions can be summarized as:

- We propose MEgo2Vec (embed the Matched Ego network to a vector), a novel graph neural network model, to formalize our problem as a unified optimization framework.
- The multi-view node embedding can model the literal and semantic characteristics of different attributes unitedly; the attention mechanism can distinguish the effects of different neighbors to alleviate error propagations; structure embedding can capture the influence of different topologies.
- Experiments on a number of academia and social networking datasets demonstrate that the proposed model compares favorably classification accuracy (+3.12-30.57% in terms of F1 score) against the state-of-the-art models for user alignment task.

2 RELATED WORK

User Alignment. One intuitive way to align users across social networks is to compare users' attributes, among which name is the most important one. Many attempts have been made to study how to leverage user names [11, 13, 17, 24, 25]. For example, Zafarani et al. dealt with user names by adding or removing their prefix/postfix [24]. They further made a comprehensive study of the habits when users select names [25]. For other attributes like the post contents of users, Kong et al. [7] calculated the cosine similarity of the bag-of-words vectors weighted by TF-IDF.

However, user attributes may be missed or not strong enough to indicate a user's identity. Thus a lot of research utilizes the network structure to enhance the prediction performance. Some of them calculated the metrics like Jaccard's coefficient or Adamic/Adar based on the number of the shared neighbors [7, 30]. However, most of the neighbor pairs are unlabeled, preventing us from calculating the above metrics. Some methods such as IONE [12] and PALE [15] learned user embeddings by encoding the structure information of the network but totally ignored user attributes, and it is not easy to add user attributes into their models.

Several research incorporates both user attributes and unlabeled neighbor pairs together. For example, Lacoste et al. solved the problem by an unsupervised propagation method [9], where the similarities of the attributes in the method are defined empirically. Zhong et al. proposed an unsupervised co-training algorithm to incorporate attributes and relationships with the neighbor pairs together [29]. Zhang et al. further formulated the propagation process based on attributes and network topologies from an optimization perspective [27], but only one attribute can be considered in their model. Zhang et al. proposed a supervised model—COSNET to iteratively infer the labels of the unlabeled pairs and update the model based on the inferred labels and the user attributes [28]. However, error propagations may be introduced in above methods. This paper aims at investigating a supervised method that

avoids label propagation, but directly represents a pair of users by their attributes, their neighbors' attributes and the network structure among the neighbors, and then aligns the users based on the learned representations.

Neural Networks on Graphs. Substantial research has been done on modeling the graph-structured input by neural networks. Shallow models such as DeepWalk [18], LINE [21] and node2vec [5] learn node embeddings via the prediction of the first-order or high-order neighbors. Graph neural networks (GNNs) were proposed as a generalization of the recursive neural networks, which assign a state to each node in a graph based on the states of neighboring nodes [4, 19]. Recently, Li et al. improved GNNs using gated recurrent units and also extended the output to sequences [10].

Several works studied how to generalize convolutions to graph data [1, 3, 6, 16, 23]. For example, Niepert et al. [16] generated a representation for a selected sequence of nodes that covers large parts of a given graph, and then used CNN to predict the label of the graph. Duvenaud et al. defined convolutions directly on graphs and learned node degree-specific weight matrix. [3]. Thomas et al. considered a single weight matrix per layer and dealt with varying node degrees by an appropriate normalization of the adjacency matrix [6]. Veličković et al. [23] introduced the attention mechanism into the graph convolutional networks to distinguish the influence of the neighbors. Inspired by these recent works, we propose a graph neural network model to align users in two networks. Most of the recent work on graph convolutional networks feed the whole graph as the input, which hinders the mini-batch training process. The proposed method in this paper, on the contrary, is performed on ego networks of two nodes, which is able to adapt to mini-batch training and thus can be conducted efficiently.

3 PROBLEM DEFINITION

In this section, we will provide necessary definitions and then formally formulate the problem. Let $G = (V, \mathbf{A}, \mathbf{P})$ denote an undirected attribute augmented social network[3], where V is set of users. Notation $\mathbf{A}$ represents a $|V|\times|V|$ adjacency matrix with an element a_{ij} indicating there is a relationship between v_i and v_j. Notation $\mathbf{P}$ represents a $|V| \times d$ property (attribute) matrix with an element p_{ik} indicating the k-th attribute of the i-th user in G.

Definition 3.1. **Ego Network.** Given a user $v_i \in V$, we use $G_i = (V_i, \mathbf{A}_i, \mathbf{P}_i) \subseteq G$ to denote v_i's ego network, where $V_i \subseteq V$ contains the focal node v_i (i.e., the node in the center) and v_i's first-order neighbors in G. Notation $\mathbf{A}_i \subseteq \mathbf{A}$ is the adjacency matrix representing the relationships between the users in V_i and $\mathbf{P}_i$ represents an $|V_i| \times d$ attribute matrix of the users in V_i.

We only consider first-order neighbors, out of the consideration of the computational efficiency and the assumption that high-order neighbors usually provide weak evidence on identifying a user. We assume that two users are more likely to be the same person, if there are more neighbors in their respective ego networks that have a high possibility to be matched as the same person. Based on the intuition, we define matched ego network as:

[3]We simply ignore the direction of a directed relationship as it is found to be not very useful in the task.

Definition 3.2. **Matched Ego Network.** Suppose we have a source network G^s and a target network G^t. Given two users $v_i \in G^s$ and $u_i \in G^t$, we denote v_i's ego network as G_i^s and u_i's ego network as G_i^t. Then we construct a matched ego network $G_i^M = (V_i^M, \mathbf{A}_i^M, \mathbf{P}_i^M)$ with each node in it as a pair of users from the two ego networks by the following steps:

(1) Add the pair of users (v_i, u_i) into V_i^M as the focal pair.
(2) Find all the potential matched neighbors of v_i and u_i, and add them into V_i^M as the neighbor pairs (More explanations are as follows).
(3) Add an edge between (v_i, u_i) and each neighbor pair. Additionally, add an edge between two neighbor pairs (v_j, u_j) and (v_k, u_k) if there is an edge between v_j and v_k in G^s, and also an edge between u_j and u_k in G^t. Notation $\mathbf{A}_i^M$ is the adjacency matrix of G_i^M.
(4) Notation $\mathbf{P}_i^M$ represents an $|V_i^M| \times 2d$ attribute matrix with each row $\mathbf{p}_j$ as a concatenation of v_j's attributes $\mathbf{p}(v_j)$ and u_j's attributes $\mathbf{p}(u_j)$, i.e., $\mathbf{p}_j = (\mathbf{p}(v_j), \mathbf{p}(u_j))$.

We will explain what are the potential matched neighbors, and why and how they are constructed. The potential matched neighbors are the neighbor pairs that have some possibility to be matched, and they are filtered from all possible pairs of the neighbors. Due to the various purposes of different social media platforms, the neighbors in two ego networks may be very diverse. Directly using all pairs of neighbors as evidence to align users may introduce much noise. Thus we propose an attention based mechanism to distinguish different neighbor pairs in our model (See details in Section 4). To improve the computational efficiency, before resorting to the model, we can easily select the most useful neighbor pairs by heuristic rules [28]. This paper simply selects the neighbor pairs if their names are similar, i.e., the Jaro-Winkler similarity of the two names is larger than a threshold, as the potential matched pairs. We also restrict the neighbor pairs to satisfy the one-to-one mapping constraint as Kong et al. did in [7], based on the same assumption that a user usually maintains one major account in each social media platform. Thus the constructed matched ego network has no more than $\min\{|V_i^s|, |V_i^t|\}$ user pairs. We use $\mathcal{N}_i^M$ to represent the set of the neighbor pairs of (v_i, u_i) including (v_i, u_i) itself.

Problem 1. ***User Alignment across Social Networks:*** *Given a set of matched ego networks $\{G_i^M = (V_i^M, E_i^M)\}$ constructed from a pair of a source network G^s and a target network G^t, we aim at learning a predictive function*

$$f : G_i^M = (V_i^M, E_i^M) \rightarrow y_i \tag{1}$$

to predict whether v_i and u_i belong to the same person in the real world, where y_i is a binary value, with 1 indicating v_i and u_i are the same person, and 0 otherwise.

4 METHODOLOGY

4.1 Overview

We argue that the above defined classification performance can be improved if the node attributes and the structure of the matched ego network are elaborately represented. The overview process of the proposed method is shown in Figure 2.

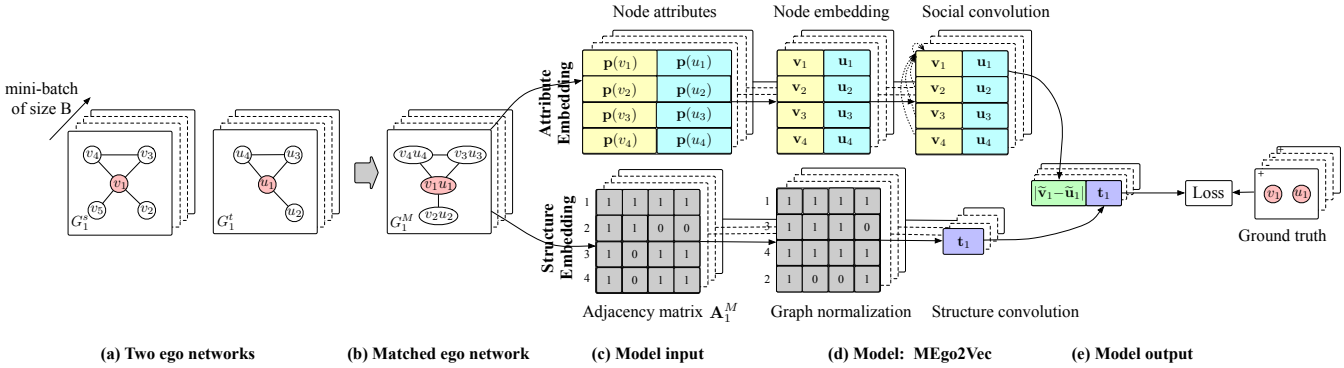


Figure 2: The Overall process of the proposed method. (a) Extract the ego networks of the two users to be predicted; (b) construct a matched ego network for the two users; (c) input the attributes of all the nodes and the adjacency matrix of the matched ego network to the model; (d) embed the attributes and the structure information by MEgo2Vec; (e) output an embedding for the matched ego network.

Candidate Generation. In practice, the number of all the user pairs across two networks is very large. Thus, we need to generate candidate user pairs from all the pairs. Following [17, 28], we only keep the user pairs if their names are similar to each other (Cf. Section 5 for the details).

Matched Ego Network Construction. For each candidate pair (v_1, u_1), we extract their ego networks G_1^s and G_1^t from their respective networks (Figure 2(a)), and construct a matched ego network G_1^M according to definition 3.2 (Figure 2(b)). Figure 2 is corresponding to the case in Figure 1, where v_2, v_3 and v_4 are matched to u_2, u_3 and u_4 respectively. User v_5 in G^s is not matched to any user in G^t, so it is removed from G_1^M. Note that a matching between two users only indicates that they have the potential possibility to be a same person, but the true label is usually unknown.

Matched Ego Network Embedding. We take the attributes of all the node pairs in G_1^M and the adjacency matrix of G_1^M as the input of the proposed model—MEgo2Vec whose output is an embedding. The model consists of two components: *attribute embedding* and *structure embedding*. The component of attribute embedding adopts the attributes as input, and generates two embeddings $\widetilde{\mathbf{v}}_1$ and $\widetilde{\mathbf{u}}_1$ for v_1 and u_1. Specifically, we first generate an embedding vector for each user in G_1^M based on his/her attributes by the proposed multi-view node embedding mechanism (Figure 3), and then update the embeddings of the focal pair (v_1, u_1) through convolving their neighbors' embeddings by the proposed attention mechanism (Figure 4). The component of the structure embedding adopts the adjacency matrix $\mathbf{A}_1^M$ as input, normalizes and convolves $\mathbf{A}_1^M$ into another embedding $\mathbf{t}_1$ (Section 4.3).

Finally, we predict the label of (v_1, u_1) based on the combination of the attribute embeddings and the structure embedding, and calculate the loss according to the ground truth. We will explain the details of different components as follows.

4.2 Attribute Embedding

Node Embedding. Many works demonstrate the effectiveness of CNN in representing texts [14]. However, in our task, there are multiple attributes for each user, and different attributes present different characteristics when they are used to determine whether two users are the same person or not. For example, in our datasets, more than 70% users change a few characters in their nicknames on different social media platforms. In this case, although the whole words of their names are totally different, the characters in those names are highly correlated. But for other attributes of long texts such as research interests, words may make more effects than characters. To model both the literal and semantic characteristics of the attributes unitedly, we propose a multi-view node embedding strategy. We will introduce the details as below.

We divide each attribute into a list of words. For each word in an attribute, we firstly represent it as an embedding vector, named as word-view embedding. Then we divide the word into a list of characters, represent each character as an embedding vector, convolve all the character embeddings by a CNN network and output an embedding as the word's character-view embedding. The word-view and character-view embeddings are concatenated as the final embedding for a word. Then the sequence of the word embeddings in an attribute are convolved by another CNN network to output the final embedding for the attribute. Finally, we use an attribute pooling strategy to aggregate the embeddings of all the attributes of a node together as the node embedding:

$$\mathbf{v}_i = \sum_{k=1}^{d} \mathbf{a}_{ik}\gamma_k, \tag{2}$$

where $\mathbf{a}_{ik}$ represents the embedding of the k-th attribute of user v_i, $\gamma_k \in \mathbb{R}$ is the corresponding weighting parameter to be learned, d is the number of attributes of a user and $\mathbf{v}_i$ is the K-dimension embedding learned for user v_i. Figure 3 illustrates the above multi-view node embedding mechanism, where all the attributes are dealt with in the same way as name.

Social Convolution. To determine whether two users can be aligned or not, the missing attributes or the attributes that are not distinguishable enough usually cannot provide sufficient evidence. But additional evidence may be obtained from the attributes of the neighbors. Thus after we obtain the embedding for each node

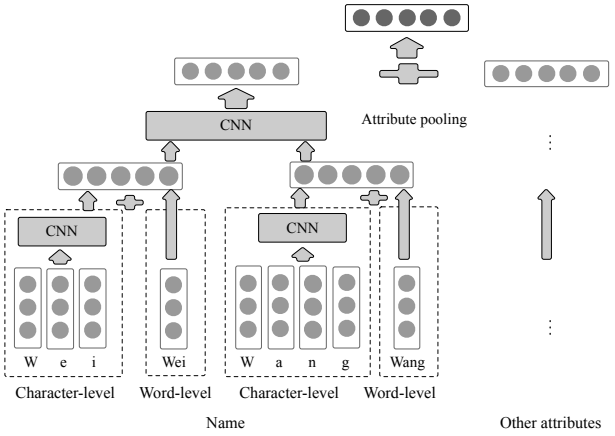


Figure 3: Multi-view node embedding.

based on their own attributes, we further smooth each node's embedding by their neighbors' embeddings.

The general idea of social convolution is that, for the focal pair (v_i, u_i) to be predicted, we convolve the above obtained embeddings of all its neighbor pairs $\{(\mathbf{v}_j, \mathbf{u}_j) : j \in \mathcal{N}_i^M\}$ (including the embeddings of the focal pair itself, i.e., $(\mathbf{v}_i, \mathbf{u}_i)$) into new embeddings of the focal pair:

$$(\widetilde{\mathbf{v}}_i, \widetilde{\mathbf{u}}_i) = c(\{(\mathbf{v}_j, \mathbf{u}_j) : j \in \mathcal{N}_i^M\}), \tag{3}$$

where $\widetilde{\mathbf{v}}_i, \widetilde{\mathbf{u}}_i \in \mathbb{R}^K$. The convolution operation $c(\cdot)$ can be defined in various ways. A straightforward way to define $c(\cdot)$ is to average the embeddings of all the neighbors together, i.e.,

$$\widetilde{\mathbf{v}}_i = \frac{1}{|\mathcal{N}_i^M|} \sum_{j \in \mathcal{N}_i^M} \mathbf{v}_j, \quad \widetilde{\mathbf{u}}_i = \frac{1}{|\mathcal{N}_i^M|} \sum_{j \in \mathcal{N}_i^M} \mathbf{u}_j. \tag{4}$$

However, different neighbor pairs may exert different effects. For aligning two users, some neighbor pairs may provide strong evidence, while others may bring in additional noise. Although we have already filtered out most of the neighbors pairs if their names are totally different when constructing the matched ego network for a focal pair, the rule to filter neighbor pairs is heuristic and is very likely to keep the wrongly matched neighbor pairs in the matched ego network. Thus, it is necessary to distinguish the effects from different neighbor pairs. Recently, the attention mechanism used in neural networks is a well-adopted way to achieve the goal [2]. Inspired by this, we propose using attention mechanisms to realize the social convolution operation $c(\cdot)$.

Essentially, attention mechanism is introduced to determine the contribution of a neighbor pair to the focal pair. Formally, given the embeddings of a neighbor pair $(\mathbf{v}_j, \mathbf{u}_j)$ and the embeddings of the focal pair $(\mathbf{v}_i, \mathbf{u}_i)$, we perform a shared attentional mechanism $g : \mathbb{R}^K \times \mathbb{R}^K \times \mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}$ on the four input embedding vectors to calculate the *attention coefficient*:

$$e_{ji} = g(\mathbf{v}_j, \mathbf{u}_j, \mathbf{v}_i, \mathbf{u}_i), \tag{5}$$

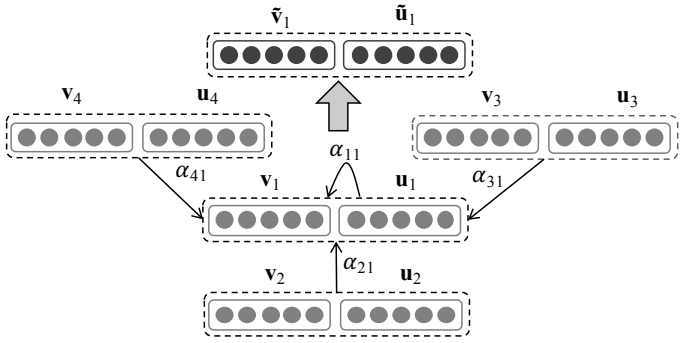


Figure 4: Social convolution. We convolve the embeddings of v_1 to v_4 by attention coefficients α_1 to α_4 to obtain the final embedding $\widetilde{\mathbf{v}}_1$ for user v_1. The same is for user u_1.

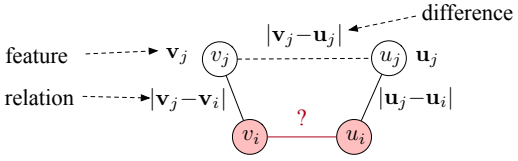


Figure 5: Illustration of different attention mechanisms that determine the contribution of a neighbor pair (v_j, u_j) on predicting the label of a focal pair (v_i, u_i).

where e_{ji} indicates the contribution of $(\mathbf{v}_j, \mathbf{u}_j)$ on predicting whether v_i and u_i are the same person or not. Then we apply the softmax function to obtain the normalized coefficients across all the neighbor pairs of (v_i, u_i):

$$\alpha_{ji} = \text{softmax}_j(e_{ji}) = \frac{\exp(e_{ji})}{\sum_{j' \in \mathcal{N}_i^M} \exp(e_{j'i})}. \tag{6}$$

The normalized coefficients are used to compute a linear combination of the embedding vectors corresponding to them. Then we apply a non-linear function $\sigma(\cdot)$ on the linear combination, which is regarded as a final embedding vector for a user:

$$\widetilde{\mathbf{v}}_i = \sigma\Big(\sum_{j \in \mathcal{N}_i^M} \alpha_{ji} \mathbf{v}_j\Big), \quad \widetilde{\mathbf{u}}_i = \sigma\Big(\sum_{j \in \mathcal{N}_i^M} \alpha_{ji} \mathbf{u}_j\Big). \tag{7}$$

Figure 4 illustrates the social convolution process. We design the attention mechanisms based on the knowledge in $\mathbf{v}_j$, $\mathbf{u}_j$, $\mathbf{v}_i$ and $\mathbf{u}_i$. Three specific attention mechanisms are proposed progressively by considering individual features of each user in a neighbor pair (*feature attention*), the similarity between two users in a neighbor pair (*difference attention*) and the relationship of a neighbor pair with the focal pair (*relation attention*).

- *Feature Attention.* The assumption of feature attention is that a neighbor pair (v_j, u_j) makes more contribution on inferring the label of the focal pair (v_i, u_i) if the features of v_j and u_j are more discriminative. For example, the neighbor pair with the same name "Jaspher Wang" benefits more on predicting the label of (v_i, u_i) than the neighbor pair with the same name "Wei Wang", as the name "Jaspher Wang" is

more unique than "Wei Wang". According to this assumption, we calculate the attention coefficient based on the concatenation of the embeddings of the neighbors in a pair:

$$e_{ji} = \mathbf{W}^T[\mathbf{v}_j; \mathbf{u}_j] + b, \tag{8}$$

where notations $\mathbf{W}$ and b in this and the following equations are the model parameters to be learned. The concatenation of the embeddings reserves the features of the two users.

- *Difference Attention.* The assumption of difference attention is that a neighbor pair (v_j, u_j) takes a more important role on predicting the label of (v_i, u_i) if the two neighbors v_j and u_j are more similar to each other. Under this assumption, we calculate the attention coefficient solely based on the difference between the embeddings of the neighbors:

$$e_{ji} = \mathbf{W}^T|\mathbf{v}_j - \mathbf{u}_j| + b, \tag{9}$$

where we use the difference between two embeddings to represent whether two neighbors are similar or not.

- *Relation Attention.* The assumption of relation attention is that a neighbor pair (v_j, u_j) takes more effects on determining the label of (v_i, u_i) if the relationship between user v_j and v_i in G^s share the same semantics with the relationship between user u_j and u_i in G^t. For example, if we already know user x in Twitter is aligned to user a in MySpace, and user y in Twitter is aligned to user b in MySpace, suppose x shares the same interest with y in Twitter, then most probably a also shares the same interest with b in MySpace. Under this assumption, we calculate the attention coefficient by comparing the difference between the embeddings of two users in two networks:

$$e_{ji} = \mathbf{W}^T||\mathbf{v}_j - \mathbf{v}_i| - |\mathbf{u}_j - \mathbf{u}_i|| + b, \tag{10}$$

where we use the difference between two users, e.g., $|\mathbf{v}_j - \mathbf{v}_i|$, to represent the semantics of the relationship between the users v_j and v_i, and then use the difference between the relationship semantics to represent whether the two relationships are similar or not.

Figure 5 illustrates the knowledge that is used to infer the attention coefficient of (v_j, u_j) on (v_i, u_i). We can also consider the knowledge used in all the three attention mechanisms and unify them in one equation:

$$e_{ji} = \mathbf{W}^T[\mathbf{v}_j; \mathbf{u}_j; |\mathbf{v}_j - \mathbf{u}_j|; ||\mathbf{v}_j - \mathbf{v}_i| - |\mathbf{u}_j - \mathbf{u}_i||] + b. \tag{11}$$

The unified attention mechanism is used in our final model.

4.3 Structure Embedding

Different from attribute embedding, the component of structure embedding aims at embedding the structure information of the matched ego network to predict the label of the focal pair. The intuition is two users are more likely to be the same person if they have more potential matched neighbor pairs, and furthermore, if there are more neighbors know each other. For example, in Figure 2(b), the focal pair (v_1, u_1) has three potentially matched neighbor pairs, which provides significant evidence on predicting the label of (v_1, u_1). In addition, neighbor pair (v_4, u_4) is linked to neighbor pair (v_3, u_3), i.e, v_4 knows v_3 and u_4 knows u_3, which further improves the probability that v_1 and u_1 are the same person. The links between neighbor pairs reflect the structural characteristics of the matched ego network, which can benefit the predicting task. From the example, we can see that the structural roles of different neighbor pairs are different when they are used to infer the label of a focal pair. The core problem of leveraging the structure information is to map the network structure to an embedding vector such that the neighbor pairs with similar structural roles in different matched ego networks are positioned similarly in the corresponding embedding vectors. Inspired by [16], we address the problem by firstly normalizing the adjacency matrix of the input matched ego network and then applying a convolutional neural network on the normalized graph to learn an embedding vector for the graph.

Normalization. Normalization aims at finding an order of the neighbor pairs in each matched ego network to reserve the relative position of the neighbor pairs. The basic idea is two neighbor pairs in different matched ego networks can be located at the similar position of the respective adjacency matrix if their structural roles within the matched ego networks are similar. Intuitively, a neighbor pair can be ranked higher if it is more closer to the focal pair. Multiple similarity metrics defined on graphs can be used to measure the distance between two nodes such as common neighbors, Jaccard Coefficient, SimRank, etc.

Structure Convolution. After we obtain a ranked list of node pairs in G_i^M, we get a new adjacency matrix $\hat{A}_i^M$ with the indexing of the node pairs correspond to the ranked pairs. Then we apply a convolutional neural network on the normalized adjacency matrix $\hat{A}_i^M$ to learn an embedding vector for the adjacency matrix. Specifically, we firstly pad dummy node pairs to the adjacency matrix if the size of the pairs is smaller than a predefined number L, and discard the pairs with indexing larger than L; then we reshape the adjacency matrix to a $L \times L$-dimension vector, and apply H different $L \times L$-dimension convoluational kernels with 1 stride, to output H-dimension embedding vector $\mathbf{t}_i$.

4.4 Objective Function

For each focal pair (v_i, u_i), we concatenate the difference between their attribute embeddings $\widetilde{\mathbf{v}}_i$ and $\widetilde{\mathbf{u}}_i$, i.e., $|\widetilde{\mathbf{v}}_i - \widetilde{\mathbf{u}}_i|$ and the structure embedding $\mathbf{t}_i$. Then we apply a function f such as the full connection operation on the concatenation to predict the matching score $\hat{y}_i = f(|\widetilde{\mathbf{v}}_i - \widetilde{\mathbf{u}}_i|; \mathbf{t}_i)$. Finally we define the objective function as the cross entropy loss of the true label and the predicted score:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \text{CrossEntropy}(f(|\widetilde{\mathbf{v}}_i - \widetilde{\mathbf{u}}_i|; \mathbf{t}_i), y_i), \tag{12}$$

where n is the total number of labeled user pairs.

Table 1: Statistics of the datasets.

Dataset	Network	#Users	#Relationships	Relation Type
Academia	Aminer	1,053,188	3,916,907	Co-author
	LinkedIn	2,985,414	25,965,384	Co-view[7]
	VideoLectures	11,178	786,353	Co-attendance[8]
SNS	Twitter	40,171,624	1,468,365,182	follow
	MySpace	854,498	6,489,736	friend

5 EXPERIMENTS

5.1 Experimental Settings

Datasets. We collect three **Academia** networks and two **SNS** networks[4]. **Academia** consists of networks collected from Aminer [22], an academic searching and mining service, LinkedIn, a business- and employment-oriented professional social network, and VideoLectures, the world's biggest academic online video repository. **SNS** consists of Twitter, a well-known microblogging social platform, and MySpace, a social networking website for users to share blogs, photos and music. The details are summarized in Table 1. We treat all the relationships in the networks as undirected ones in our experiments. All data and codes are available online[5].

Training Data. An instance is a pair of users from two different networks, where a positive instance means the two users belong to a same person, while a negative instance indicates they are different persons. For the training data constructed across SNS networks, the positive instances were collected by Perito el. al using Google Profile Service (GPS) [17]. GPS[6] allows users to provide their accounts on different online social networks, and the accounts from same users can be treated as positive instances. For those constructed across academia networks, the positive instances were provided by the users in Aminer.org, where users can input the links of their LinkedIn and ViedoLectures profile pages in their Aminer profile pages. Negative instances should be constructed as they cannot be directly collected by existing services. Following the assumption that a user usually maintains one major account in each platform [7], we get that in most cases the users appeared in positive instances cannot be aligned to others users. Thus we sample several negative instances for each positive instances. Specifically, for each positive instance (v_i, u_i) where v_i is from G^s and u_i is from G^t, we randomly sample several negative partners from G^t for v_i, and several negative partners from G^s for u_i. All the negative instances are sampled from the generated candidate pairs (Cf. Section 4.1). Finally, we keep the ratio between positive and negative instances as about 1:10 and collect 33,981, 34,060 and 35,080 training instances for Aminer-LinkedIn, Aminer-VideoLectures and Twitter-MySpace respectively.

For all the users in training data of Academia networks, we select the profile attributes of name, affiliation, education and research interests/skills to compare. Although there are many attributes in SNS networks, most of the attributes are extremely sparse. Thus we only choose screen name, account name and locations to compare for users in the training data of SNS networks.

Baseline Methods. The comparison methods include:

Exact Name Match(ENM): considers a user pair collected from two networks as a positive instance if the name of a user is an exact match to that of the other user in the pair. A user in one network may match to multiple users with same names in the other network.

SVM: defines the features as the Jaro-Winkler similarity of two names, the edit distance of two names, and the cosine similarity between the TF-IDF weighted word vectors of other attributes.

SVM+N: averages the values for each feature in SVM over all the neighbor pairs in a matched ego network as additional features.

SiGMa [9]: begins from a seed set of user pairs (output by Name-Match and then filtered out the most common names) and iteratively augments the seed set by propagating the matching scores (predicted by SVM) through the two input networks.

COSNET [28]: is a factor graph model that incorporates the attributes of two users as local factors (the same as SVM), and the relationships between user pairs as correlation factors. The intuition is two linked user pairs tend to be have the same labels.

MEgo2Vec: is the proposed model that incorporates the multi-view node embedding, unified attention mechanism and the structure embedding together.

Variants of Our Model. We try seven variants of MEgo2Vec to evaluate the effectiveness of different components one by one.

Multi-view Node Embedding: Firstly, to evaluate the effect of the multi-view node embedding component, we ignore the components of attention based social convolution and structure embedding, and only embed the attributes of the focal pair itself. The model is denoted as MEgo2Vec_{cw}. Additionally, we try word-view and character-view embeddings separately, and denote them as MEgo2Vec_{w} and MEgo2Vec_{c} respectively.

Social Convolution: Secondly, based on MEgo2Vec_{cw}, we add the attribute embeddings of neighbor pairs to evaluate the effect of the attention based social convolution component. The model utilizes the unified attention mechanism and ignores the structure embedding, and is denoted as MEgo2Vec_{u}. Additionally, we try the feature attention, difference attention and relation attention mechanisms separately, and denote them as MEgo2Vec_{f}, MEgo2Vec_{d} and MEgo2Vec_{r} respectively. We also simply average the attribute embeddings of neighbor pairs and denote it as MEgo2Vec_{a}, i.e., attention coefficient α_{ji} is set as $1/\mathcal{N}_i^M$.

Structure Embedding: Finally, based on MEgo2Vec_{u}, we add the structure embedding result to evaluate the effect of the structure embedding component. The model is denoted as MEgo2Vec and is also the final proposed model.

Evaluation Strategies. We evaluate all the comparison methods in terms of Precision, Recall and F1-Measure.

Implementaion Details. We implement the model by Tensorflow and run the code on an Enterprise Linux Server with 6 Intel(R) Xeon(R) CPU cores (E5-2699 and 56G memory) and 1 NVIDIA Tesla P40 GPU core (24G memory).

[4]https://www.aminer.cn/cosnet
[5]https://github.com/BoChen-Daniel/MEgo2Vec
[6]http:www.google.comprofiles

[7]Co-view means two user profiles were viewed by a same user.
[8]Co-attendance means two users attended a same event.

Table 2: Prediction performance on three datasets (%).

Method	Aminer-LinkedIn			Aminer-VideoLectures			Twitter-MySpace		
	Pre.	Rec.	F1	Pre.	Rec.	F1	Pre.	Rec.	F1
ENM	59.50	69.94	64.30	46.56	84.10	59.54	73.29	41.82	53.25
SVM	95.97	76.17	84.93	86.96	72.80	79.25	98.40	56.05	71.42
SVM+N	95.78	81.62	88.14	86.71	76.09	81.06	97.74	60.58	74.80
SiGMa	88.50	47.39	61.72	87.56	58.10	69.85	88.29	35.82	50.96
COSNET	84.84	83.15	83.98	69.19	**87.10**	77.10	78.48	69.56	73.75
MEgo2Vec	**95.58**	**89.21**	**92.29**	**89.29**	79.62	**84.18**	**88.47**	**72.95**	**79.97**

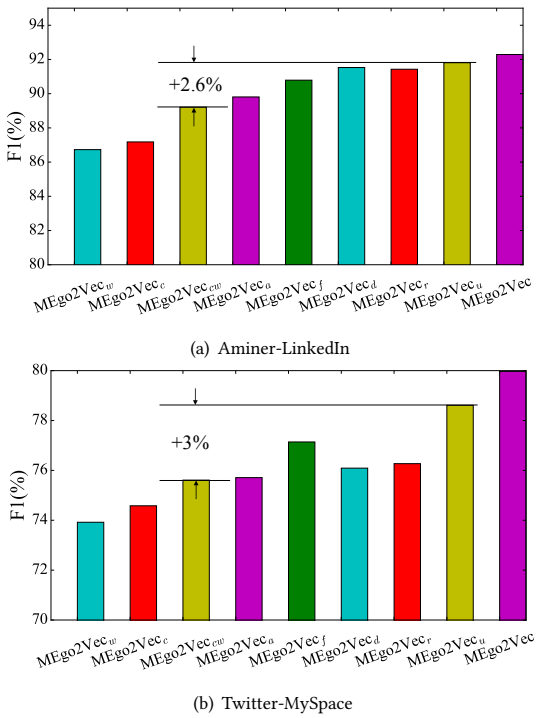
Table 3: Prediction performance on the instances of Aminer-LinkedIn with or without neighbors by MEgo2Vec$_{cw}$ and MEgo2Vec$_u$ (%).

Instances	Method	Precision	Recall	F1
With Neighbors	**MEgo2Vec$_{cw}$**	99.35	85.56	91.94
	MEgo2Vec$_u$	99.10	92.22	95.54
Without Neighbors	**MEgo2Vec$_{cw}$**	86.03	84.16	85.09
	MEgo2Vec$_u$	88.99	82.43	85.59

For the academia networks, we select the candidate pairs if they share the same firstname, and the same initials of the lastname. For the SNS networks, we select the candidate pairs if their screen names or account names share at least four 2-grams. When generating a matched ego network, the threshold to filter neighbor pairs is set as 0.8. In node embedding component, the dimension of word, character and attribute embedding is set as 64, 32 and 192 respectively. In social convolution component, the maximal number of neighbor pairs in a matched ego network is restricted to 14. Then we pad dummy neighbor pairs if the size of the pairs is smaller than 14, and discard additional pairs otherwise. In structure embedding component, we use common neighbors as the similarity metric to conduct graph normalization. Notation L denotes the maximal number of neighbor pairs, and is also set as 14. The dimension H, i.e., the output channel of CNN network, is set as 64. The learning rate of the model is set as 0.001. The size B of a mini-batch is 100.

5.2 Performance Analysis

Overall Prediction Performance. Table 2 shows the overall performance on three pairs of networks. The proposed method performs clearly better than the comparison baselines on all datasets (+3.12-30.57% in terms of F1 score). Our method improves more F1 on the two academia datasets than on the SNS dataset, as the attributes of the academia dataset are much richer than those of the SNS dataset, which indicates neural networks are more suitable for the complex input dataset. ENM is essentially a rough rule and performs poorly, as on one hand, it totally ignores the positive instances with different names, and on the other hand, different users with same names are wrongly identified as the same users. SVM ignores the network information and defines features by human experience, which is not comprehensive enough to represent the attributes. SVM+N leverages the attributes of neighbors, but it



(a) Aminer-LinkedIn

(b) Twitter-MySpace

Figure 6: Prediction performance of model variants.

does not distinguish the effects of different neighbors. SiGMa results in high precision but suffers from a low recall. Because the overlap between the input two networks is low, leading it not easy to propagate the matching scores from the seeds to other pairs. COSNET essentially propagates the inferred labels in the whole matched network, thus it may suffer from error propagation. Besides, the features in COSNET is also human defined. MEgo2Vec uses a graph neural network to embed the neighbors' attributes and the structure information, which can capture both the literal and semantic characteristics of the attributes, and meanwhile alleviate error propagations through the network.

On the academia datasets, processing 100 mini-batches requires 150 seconds, and best validation F1 is reached after 20-40 epoches (about 270 mini-batches per epoch) through the data, which requires 2-3 hours of training. On Twitter-MySpace, as the attributes are extremely sparse, i.e., features are not sufficient, processing 100 mini-batches requires 25 seconds, but best validation F1 is reached after 100-120 epoches (281 mini-batches per epoch), which requires 2 hours of training. We show the prediction performance of the different variants of our model on one academia dataset and one SNS dataset in Figure 6 and analyze the results as follows.

Multi-view Node Embedding Effect. We compare two single view models—MEgo2Vec$_w$ and MEgo2Vec$_c$ with the multi-view model MEgo2Vec$_{cw}$, and show the results in Figure 6. We see that MEgo2Vec$_{cw}$ performs better than MEgo2Vec$_w$ (+1.69-2.48% in F1) and MEgo2Vec$_c$ (+1.03-2.03% in F1), which demonstrates the strength of the multi-view embedding mechanism when representing the

node attributes. MEgo2Vec$_c$ performs better than MEgo2Vec$_w$, as the similarity between the characters of names takes the most important role among all the attributes. Compared with MEgo2Vec$_w$, MEgo2Vec$_c$ improves more F1 on Twitter-MySpace than on Aminer-LinkedIn, as other attributes except name are extremely sparse on the SNS dataset, thus characters in names dominate the effect.

Neighbor Pair Effect. To verify the effect of social convolution component, based on MEgo2Vec$_{cw}$, we add the neighbors' attribute embeddings by the unified attention mechanism and find that the model MEgo2Vec$_u$ performs much better than MEgo2Vec$_{cw}$ (+2.60-3.00% in F1). We further evaluate the instances with and without neighbors separately on Aminer-LinkedIn and show the results in Table 3. We can see that if predicted by MEgo2Vec$_u$ instead of MEgo2Vec$_{cw}$, the performance is improved by 3.6% in F1 for the instances with neighbors, but it is almost not improved for those without neighbors. The results indicate that the neighbors' information captured by our method can indeed benefit the task. Note that in the instances with and without neighbors, the ratio between the positive and negative parts is different, so the performance is different by the same method.

Social Convolution Effect. We compare the three attention mechanisms MEgo2Vec$_d$, MEgo2Vec$_f$ and MEgo2Vec$_r$ and show the results in Figure 6. MEgo2Vec$_r$ and MEgo2Vec$_d$ perform better than MEgo2Vec$_f$ on academic networks, as relation attention and difference attention determine the contribution of a neighbor pair by comparing two neighbors' difference or two relations' difference, while feature attention only considers the features of a user, but ignores the difference between two users. But on Twiter-MySpace, since the attributes in the SNS networks are quite sparse except name, the relationships between users are not so easy to describe, making MEgo2Vec$_f$ perform better than the other attentions. When combining the three attention mechanisms as MEgo2Vec$_u$ does, the performance is better than each individual attention mechanism. MEgo2Vec$_a$ averages neighbors' embeddings. It performs almost the same as MEgo2Vec$_{cw}$ that totally ignores neighbors' information. The results indicate that neighbor pairs may bring in noise and it is necessary to carefully utilize the neighbors' information by attention mechanism.

Structure Embedding Effect. Based on MEgo2Vec$_u$, we add the structure embedding, and denote it as our final model MEgo2Vec. The results in Figure 6 show that the performance can be improved by 0.48-1.36% in terms of F1, which indicates that the structure information of the matched ego network is effectively modeled by the structure embedding component, and can indeed benefit the alignment task. But the effect is not so significant, as the connections between neighbor pairs in the collected datasets are sparse.

5.3 Case Study

Attention Coefficients. We analyze the attention coefficients learned by the proposed attention mechanisms. Figure 7 shows a case for predicting whether two "Osmar R. Zaiane"s in two ego networks are the same person or not. We present the learned attention coefficients of each neighbor by MEgo2Vec$_r$ beside the edges. It shows that the neighbor pair named Russell Greiner makes the biggest contribution (with attention as 0.27) among all the neighbor pairs except the focal pair itself. Because Russell shares the same affiliation with Osmar, and also gets the same degree of computer science (inferred from their educations) with Osmar in the first ego network. And in the second ego network, Russell Greiner gets the same degree of computer science with Osmar, and also shares the same interest like "machine learning" and "computer

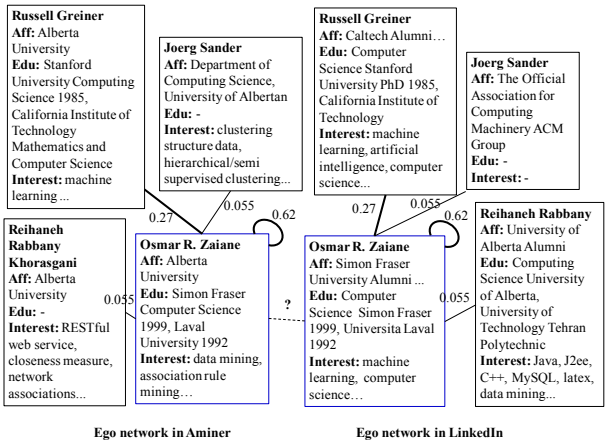
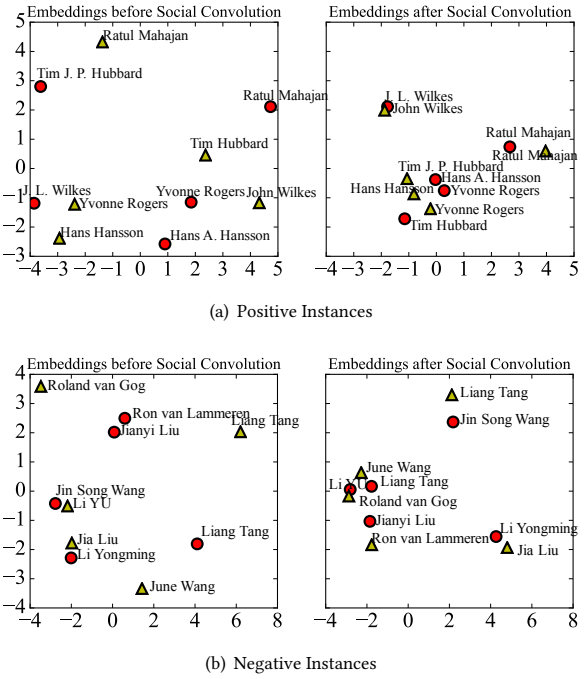


Figure 7: Case study of learned attention coefficients.



(a) Positive Instances

(b) Negative Instances

Figure 8: Case study of learned embeddings. ("Tim J. P. Hubbard", "Tim Hubbard"), ("J L. Wilkes", "John Wilkes"), ("Hans Hansson", "Hans A. Hansson"), ("Ratul Mahajan", "Ratul Mahajan"), ("Yvonne Rogers","Yvonne Rogers"), ("Jianyi Liu, "Jia Liu"), ("Liang Tang", "Liang Tang"), ("Ron van Lammeren","Roland van Gog"), ("Jin Song Wang", "June Wang") and ("Li Yongming","Li YU") are the positive and negative instances.

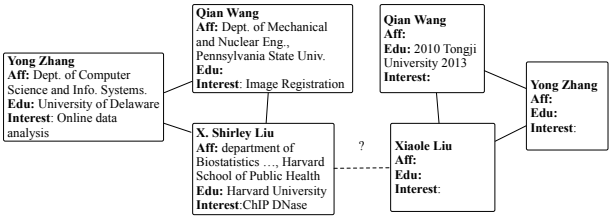


Figure 9: Case study of structure embedding component.

science" with Osmar. The relationship between Russell and Osmar in the first network is quite similar to the relationship between them in the second network. If calculated by MEgo2Vec$_d$, Russell also makes more contribution than other pairs, because the two "Russell Greiner"s share the similar educations and interests.

Node Embedding Visualization. Figure 8 visualizes the node embeddings before the social convolution operation, i.e., $(\mathbf{v}_i, \mathbf{u}_i)$, and the embeddings after the social convolution operation, i.e., $(\widetilde{\mathbf{v}}_i, \widetilde{\mathbf{u}}_i)$, by MEgo2Vec$_u$ for 5 randomly selected positive and negative instances with neighbors respectively. The circle points are the users in Aminer, and the triangle points are their partners in LinkedIn. For the positive instances in Figure 8(a), the two users in a pair become more similar to each other after the social convolution operation, but for the negative instances in Figure 8(b), the two users in a pair do not tend to be more similar after the social convolution operation, which indicates that the proposed model can effectively leverage the neighbor pairs to align users.

Sturcture Embedding. Figure 9 shows a user pair ("X. Shirley Liu, "Xiaole Liu") that is correctly predicted as the same person by MEgo2Vec, but is wrongly predicted by MEgo2Vec$_u$. The case is difficult to be predicted, because firstly, the attributes of one network is totally missing, and even the name of the two users are not very similar; secondly, although there are two potentially matched neighbor pairs ("Yong Zhang", "Yong Zhang") and ("Qian Wang", "Qian Wang"), their attributes are very sparse in one network, thus can not provide enough evidence. Fortunately, two neighbors know each other in their respective networks, which is an important clue to indicate that "X. Shirley Liu and "Xiaole Liu" are the same person. The structure information can be well captured by the structure embedding component of our model.

6 CONCLUSION

We present the first attempt to solve the problem of user alignment by a graph neural network model, which predicts whether two users can be aligned or not by matching and embedding their ego networks. The model deals with the issue of diverse neighbors by distinguishing them using attention mechanisms. Meanwhile, it captures both the literal and semantic characteristics of the attributes by a multi-view embedding mechanism, and also captures the structure of the matched ego network through normalizing and convolving the adjacency matrix. The experimental results on different genres of datasets validate the effectiveness of the model.

Acknowledgments. This work is supported by National Key R&D Program of China (No.2018YFB1004401) and NSFC under the grant No. 61532021, 61772537, 61772536, 61702522 and the Research Funds of Renmin University of China(15XNLQ06). *Hong Chen is the corresponding author.

REFERENCES

[1] James Atwood and Don Towsley. 2016. Diffusion-convolutional neural networks. In *NIPS'16*. 1993–2001.
[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR'15*.
[3] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS'15*. 2224–2232.
[4] Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. In *IJCNN'05*. 729–734.
[5] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD'15*. 855–864.
[6] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR'17*.
[7] Xiangnan Kong, Jiawei Zhang, and Philip S Yu. 2013. Inferring anchor links across multiple heterogeneous social networks. In *CIKM'13*. 179–188.
[8] Nitish Korula and Silvio Lattanzi. 2014. An efficient reconciliation algorithm for social networks. *Proceedings of the VLDB Endowment* 7, 5 (2014), 377–388.
[9] Simon Lacoste-Julien, Konstantina Palla, Alex Davies, Gjergji Kasneci, Thore Graepel, and Zoubin Ghahramani. 2013. Sigma: Simple greedy matching for aligning large knowledge bases. In *SIGKDD'13*. 572–580.
[10] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2016. Gated graph sequence neural networks. In *ICLR'16*.
[11] Jing Liu, Fan Zhang, Xinying Song, Young-In Song, Chin-Yew Lin, and Hsiao-Wuen Hon. 2013. What's in a name?: an unsupervised approach to link users across communities. In *WSDM'13*. 495–504.
[12] Li Liu, William K Cheung, Xin Li, and Lejian Liao. 2016. Aligning Users across Social Networks Using Network Embedding.. In *IJCAI'16*. 1774–1780.
[13] Siyuan Liu, Shuhui Wang, Feida Zhu, Jinbo Zhang, and Ramayya Krishnan. 2014. Hydra: Large-scale social identity linkage via heterogeneous behavior modeling. In *SIGMOD'14*. 51–62.
[14] Xuezhe Ma and Eduard Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *ACL'16*. 1064–1074.
[15] Tong Man, Huawei Shen, Shenghua Liu, Xiaolong Jin, and Xueqi Cheng. 2016. Predict Anchor Links across Social Networks via an Embedding Approach.. In *IJCAI'16*. 1823–1829.
[16] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *International conference on machine learning*. 2014–2023.
[17] Daniele Perito, Claude Castelluccia, Mohamed Ali Kaafar, and Pere Manils. 2011. How Unique and Traceable Are Usernames?. In *ICPET'11*. 1–17.
[18] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD'14*. 701–710.
[19] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *TNN* 20, 1 (2009), 61–80.
[20] Kai Shu, Suhang Wang, Jiliang Tang, Reza Zafarani, and Huan Liu. 2017. User Identity Linkage across Online Social Networks: A Review. *ACM SIGKDD Explorations Newsletter* 18, 2 (2017), 5–17.
[21] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW'15*. 1067–1077.
[22] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *SIGKDD'08*. 990–998.
[23] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR'18*.
[24] Reza Zafarani and Huan Liu. 2009. Connecting Corresponding Identities across Communities. *ICWSM'09* 9 (2009), 354–357.
[25] Reza Zafarani and Huan Liu. 2013. Connecting users across social media sites: a behavioral-modeling approach. In *SIGKDD'13*. 41–49.
[26] Reza Zafarani, Lei Tang, and Huan Liu. 2015. User identification across social media. *TKDD* 10, 2 (2015), 16.
[27] Si Zhang and Hanghang Tong. 2016. FINAL: Fast Attributed Network Alignment.. In *SIGKDD'16*. 1345–1354.
[28] Yutao Zhang, Jie Tang, Zhilin Yang, Jian Pei, and Philip S Yu. 2015. COSNET: Connecting heterogeneous social networks with local and global consistency. In *SIGKDD'15*. 1485–1494.
[29] Zexuan Zhong, Yong Cao, Mu Guo, and Zaiqing Nie. 2018. CoLink: An Unsupervised Framework for User Identity Linkage. (2018).
[30] Xiaoping Zhou, Xun Liang, Haiyan Zhang, and Yuefeng Ma. 2016. Cross-platform identification of anonymous identical users in multiple social media networks. *TKDE* 28, 2 (2016), 411–424.