Learning Binary Hash Codes for Fast Anchor Link Retrieval across Networks

Yongqing Wang^{1,2,†}, Huawei Shen^{1,2,†}, Jinhua Gao^{1,2,*} and Xueqi Cheng^{1,2,†}
[1]CAS Key Laboratory of Network Data Science and Technology,
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
†\{wangyongqing,shenhuawei,cxq\}@ict.ac.cn
*\{gaojinhua\}@software.ict.ac.cn,

ABSTRACT

Users are usually involved in multiple social networks, without explicit *anchor links* that reveal the correspondence among different accounts of the same user across networks. Anchor link prediction aims to identify the hidden anchor links, which is a fundamental problem for user profiling, information cascading, and cross-domain recommendation. Existing methods mainly focus on improving the accuracy of anchor link prediction, leveraging either user profile information or structural features extracted or learned from social networks. However, it is not straightforward to deploy an anchor link prediction method in practical system. The challenge is attributed to that anchor link prediction is a mapping problem across multiple high-dimensional spaces, which requires high computational cost when identifying the counterpart of one node in the other network. To combat this problem, in this paper we propose a novel embedding and matching architecture to directly learn binary hash code for each node. Hash codes offer us an efficient index to filter out the candidate node pairs for anchor link prediction. Extensive experiments on synthetic and real world large-scale datasets demonstrate the proposed method outperforms state-of-the-art methods at both the prediction accuracy and the efficiency in anchor link prediction.

KEYWORDS

anchor link prediction, hash code, network embedding, deep learning

ACM Reference Format:
Yongqing Wang^{1,2,†}, Huawei Shen^{1,2,†}, Jinhua Gao^{1,2,*} and Xueqi Cheng^{1,2,†} ^{1}CAS Key Laboratory of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China ^{2}University of Chinese Academy of Sciences, Beijing, China †\{wangyongqing,shenhuawei,cxq\}@ict.ac.cn *\{gaojinhua\}@software.ict.ac.cn, . 2018. Learning Binary Hash Codes for Fast Anchor Link Retrieval across Networks. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

With the benifit of socialized online services, users are often active across multiple online social networks simultaneously, e.g., Wechat and Weibo. The integration of user's data from multiple social networks can help to describe comphrehensive user profile and interests so as to provide various of applications, including precision marketing and cyber security [2, 27, 34]. Meanwhile, the transfered user data from well established social media sites can mitigate the data sparsity and solve cold start problem in new founded sites. The key of bridging user identities from multiple social networks can be formalized as *anchor link prediction* [20, 21], i.e., identifying hidden inter-network links connecting the accounts from the same users across different networks.

Existing methods for anchor link prediction mainly fall into two categories, each with their strength and weakness, according to the type of information they leverage. The first category of methods relies on the profile information of users and gains success on some specific scenarios [13, 17, 31]. However, the performance of these methods heavily depends on the availability and quality of user profile [4, 23, 28]. Consequently, these methods are difficult to be generalized to various scenarios [29, 30]. The second category of methods, in contrast, resorts to structural information, anticipating good generalization capacity. These methods extract or learn structural features from network, and leverage these features for anchor link prediction. Typical examples include exploiting global and local structural consistence [7, 8, 22, 33] and topological similarity [1, 8, 10, 12] across networks. One prominent challenge for these methods lies in how to learn a good representation for each user/node from high-dimensional and sparse raw representation of networks. Although existing methods perform rather well in the accuracy of anchor link prediction [16, 18, 32], they still suffer from big challenge when being deployed in practical systems.

The most prominent challenge for anchor link prediction lies in that anchor link prediction is a mapping problem across two high-dimensional spaces. Given a node in one network, for anchor link prediction, we need to check each node in the other network to identify its counterpart, if any.

Therefore, to combat the high computational cost, we need to learn a representation for each node that could be indexed to efficiently filter out candidate users for anchor link prediction.

In this paper, we propose a novel deep learning architecture for learning binary hash codes across networks with characteristics in low storage cost and fast retrieval speed by the means of indexing candidate users in anchor link prediction. The goal of our proposed method is to map the structural features from different networks into a Hamming space of binary codes, preserving the similarity between user identities. The compact binary codes are capable of highly efficient retrieval on large-scale datasets [5, 6, 25, 26]. We use two anchor-link-aware stages in proposed binary hash codes learning method, including embedding and matching. Firstly, we conduct network embedding on each network to capture structural features. Embedding method is compatible with learning effective representation of users in low-dimenional space. Besides, the learned network embedding preserves specific structural regularities in consideration of labeled anchor links, serving generalization of mapping from observed anchor links to hidden anchor links. In matching stage, the network embedding from different networks are mapped into a common Hamming space and matched according to the similarity measure in the mapping space. As the capability of deep neural network in capturing non-linear correlations across networks, the two stage modeling are implemented by an end-to-end deep learning architecture, where the network structures are the inputs and the binary hash codes are the outputs. More specifically, the main contributions of this work are three-folds:

- We propose a novel anchor link modeling method with learning binary hash codes according to network structures. The proposed modeling method is characteritic in low storage cost and fast retrieval speed.
- We propose a novel anchor-link-aware network embedding method and an efficient matching model based on learned user representations, which improve the prediction accuracy in anchor link prediction.
- Experiments on synthetic and large-scale real datasets show that our proposed method has consistently and significantly better performance on prediction accuary and time efficiency in comparison to alternative competitors. Moreover, we also conduct experiments on time complexity and the quality of anchor link candidate election, showing the potentials of our proposed method in real applications with large-scale networks.

2 PROBLEM DEFINITION

In this section, we introduce some basic notations and defintions in anchor link predicion task. A social network can be denoted as $G = \{\mathcal{U}, \mathcal{E}\}$, where $\mathcal{U}$ contains all user identities and $\mathcal{E} \subseteq \mathcal{U} \times \mathcal{U}$ is the set of social relationships in the network. Without loss of generality, we focus on the anchor link prediction on two social networks, denoted as network G^s

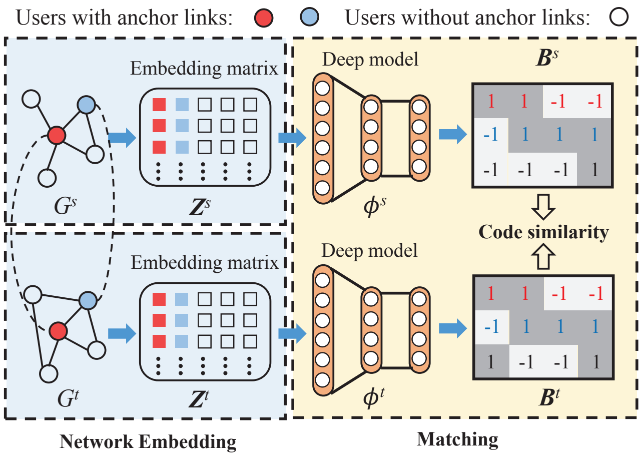


Figure 1: The framework of our proposed DHL-ALP model.

and network G^t respectively. Note that the settings of anchor link prediction on two networks can be easily extended to multiple social networks. For each user identity in one network, the objective of anchor link prediction is to identify, if any, its courterpart in another network. This task can be formally formulated as follows.

Definition 2.1 (Anchor link prediction). Given two social networks $G^s = \{\mathcal{U}^s, \mathcal{E}^s\}$ and $G^t = \{\mathcal{U}^t, \mathcal{E}^t\}$, the objective of anchor link prediction is to learn the discriminative function $\mathcal{F} : \mathcal{U}^s \times \mathcal{U}^t \to \{0, 1\}$, which estimates whether a pair of user identities from two networks belong to the same nature person, i.e.,

$$\mathcal{F}(u, v) = \begin{cases} 1,\text{if } u \in \mathcal{U}^s \text{ and } v \in \mathcal{U}^t \text{ belong to same person,} \\ 0, \textit{otherwise.} \end{cases}$$

A labeled set of anchor links, denoted as $\mathcal{A} = \{(u, v)|u \in \mathcal{U}^s, v \in \mathcal{U}^t, \mathcal{F}(u, v) = 1\}$, is also provided as the supervised information for learning function $\mathcal{F}$.

We propose a supervised **D**eep **H**ashing **L**earning framework for **A**nchor **L**ink **P**rediction with two anchor-link-aware stages, called DHL-ALP. The architecture of proposed model is shown in Fig. 1. In this model, the first stage conducts network embedding on each network respectively. The objective of network embedding is to compress the high dimensional sparse network structure (i.e., adjacent matrix) into a low dimensional dense representation, preserving social structures of users. The second stage maps network embedding from each network into a common Hamming space and identifies the anchor links in the mapping space. Formally, the objective function of our proposed model can be presented as

$$\min_{\mathbf{Z}^s, \mathbf{Z}^t, \mathbf{B}^s, \mathbf{B}^t} \mathcal{L} = O\left(G^s, G^t, \mathcal{A}|\mathbf{Z}^s, \mathbf{Z}^t\right) + \mathcal{F}\left(\mathcal{A}|\mathbf{B}^s, \mathbf{B}^t\right), \quad (1)$$

where $\mathbf{Z}^s$ and $\mathbf{Z}^t$ are the embedding matrices of network G^s and G^t respectively. The anchor-link-aware embedding loss $O\left(G^s, G^t, \mathcal{A}|\mathbf{Z}^s, \mathbf{Z}^t\right)$ reflects the capablity of generating networks by $\mathbf{Z}^s$, $\mathbf{Z}^t$ under the constaints of $\mathcal{A}$. In the second part of the objective function, the symbol $\mathbf{B}$ refers to $\mathrm{sign}(\phi(\mathbf{Z}))$ with each element being defined as follows:

$$\mathbf{B} = \mathrm{sign}(\phi(\mathbf{Z})) = \begin{cases} 1, \phi(\mathbf{Z}) \geq 0, \\ -1, \phi(\mathbf{Z}) < 0, \end{cases}$$

where the mapping functions $\phi : \mathbf{Z} \rightarrow \mathbf{X}$ try to map each user representation into a data point in $\mathbf{X}$ which can be easily transferred into a binary Hamming space. Note that the inputs and the parameters are independent in $\mathbf{B}^s$ and $\mathbf{B}^t$ respectively, we use the same notation for briefly illustrating the objective function. Therefore, the matching loss $\mathcal{F}\left(\mathcal{A}|\mathbf{B}^s, \mathbf{B}^t\right)$ depicts how well the learned binary hash codes capture the observed anchor links.

Direct optimization on Eq. (1) is quite difficult because of the interdependence between $\mathbf{Z}^s$, $\mathbf{Z}^t$, $\mathbf{B}^s$ and $\mathbf{B}^t$. Therefore, we turn to optimize the objective function by two stages separately, including network embedding and matching. Next we will introduce our proposed model in detail.

3 MODEL

In this section, we will introduce the detail of our proposed model by two anchor-link-aware stages: network embedding and matching.

3.1 Network embedding

Network embedding aims to embed a network into a low dimensional latent space, where each user is represented as a d-dimenional vector. The learned user representaions are capable of depicting neigborhood similarity and community membership, which are important features for exploring anchor links. However, under the issues of anchor link prediction, independently learning embeddings for each network may lead to great matching errors when trying to map user representations from independent embeded spaces. Therefore, we introduce an anchor-link-aware network embeddding method, exhibiting better performance in anchor link prediction.

Firstly, we introduce two vectors z_i' and z_i' for user i, where z_i is user i's representation and z_i' is her *context* representation. For each directed edge $\langle u_i, u_j \rangle$, we define the probability of *context* u_j generated by user u_i as follows,

$$p\left(u_j|u_i\right) = \frac{\exp\left(z_j'^{T} \cdot z_i\right)}{\sum_{k=1}^{|\mathcal{U}|} \exp\left(z_k'^{T} \cdot z_i\right)}, \tag{2}$$

where $|\mathcal{U}|$ is the number of users in the network. The Eq. (2) captures the similarity between two users who share common neighborhoods. According to Eq. (2), the embedding loss of network G^s and G^t can be respectively formalized as

$$O\left(G^s\right) = -\sum_{\langle u_i^s, u_j^s \rangle \in \mathcal{E}^s} \log p\left(u_j^s|u_i^s\right), \tag{3}$$

and

$$O\left(G^t\right) = -\sum_{\langle u_i^t, u_j^t \rangle \in \mathcal{E}^t} \log p\left(u_j^t|u_i^t\right). \tag{4}$$

For circumventing the performance reduction introduced by independently embeding networks, we introduce constraints to all the nodes in the labeled anchor link set. For each anchor link $(u_i^t, u_j^s) \in \mathcal{A}$, we define the probability of its generation as

$$p\left(u_i^t, u_j^s\right) \sim \frac{z_i^t \cdot z_j^s}{\|z_i^t\|\|z_j^s\|}, \; \textit{iff} \; \left(u_i^t, u_j^s\right) \in \mathcal{A}, \tag{5}$$

and the negative logarithmic likelihood of anchor link set $\mathcal{A}$ can be formulated as

$$O\left(\mathcal{A}|G^s, G^t\right) = -\sum_{(u_i^t, u_k^s) \in \mathcal{A}} \log p\left(u_i^t|u_k^s\right) \tag{6}$$

According to Eq. (3), Eq. (4) and Eq. (6), we have the anchor-link-aware embedding loss in the embedding stage, described as

$$\min_{\mathbf{Z}^s, \mathbf{Z}^t} O\left(G^s\right) + O\left(G^t\right) + \gamma_e O\left(\mathcal{A}|G^s, G^t\right), \tag{7}$$

where γ_e is a hyper-parameter.

3.2 Matching

In previous works, finding one user's counterparts in other networks needs to loop over all possible candidates, resulting in huge matching cost. In this paper, we adopt hashing method to map user representations from the originial space into a Hamming space of binary codes. The binary codes can preserve the similarity in the original space and improve retrieval speed in anchor link prediction task. Thus, we introduce an end-to-end deep architecture for efficiently learning binary codes in the maching stage.

Firstly, we learn two mapping functions on each network, transferring the network embeddings into a continuous common space so as to determine whether the user identities are matched. We define $\phi_i^s = \phi^s(z_i^s)$ and $\phi_j^t = \phi^t(z_j^t)$, referring to the outputs of mapping function ϕ^s and ϕ^t with $z_i^s \in \mathbf{Z}^s$ and $z_j^t \in \mathbf{Z}^t$ serving as inputs respectively. The matching problem is formalized as a multi-class classification issue where the inputs are the outputs of mapping functions from two networks. We define the probability of the anchor link with u_i^s and its counterpart u_j^t in two networks as

$$P\left(u_j^t|u_i^s\right) = \frac{\exp\left(\phi_j^{t\,T} \cdot \phi_i^s\right)}{\sum_{u_k^t \in \mathcal{U}^t} \exp\left(\phi_k^{t\,T} \cdot \phi_i^s\right)}. \tag{8}$$

According to the probability defined in Eq. (8), the matching loss can be formulated as follows,

$$\mathcal{L}_{match} = -\sum_{(u_i^s, u_j^t) \in \mathcal{A}} \log p\left(u_j^t|u_i^s\right). \tag{9}$$

In most senarios, the user representations learned in different networks are distinct from each other because of the independent network embedding process. Thus, we consider a non-linear mapping function, implemented by fully connected deep neural networks, to better explore the common space.

Then we learn the binary codes based on the continuous common space. We define the loss between binary codes and mapped user representations as

$$\mathcal{L}_{hash} = \|\mathbf{B}^s - \vec{\phi}^s\|_F^2 + \|\mathbf{B}^t - \vec{\phi}^t\|_F^2, \tag{10}$$

where $\vec{\phi}$ is the matrix of mapped user representations where the i-th column corresponds to user u_i. The loss limits the distance between the outputs of mapping functions and corresponding binary codes. The way of relaxation on binary codes avoids discrete learning problem, which may deteriorate the accuracy of the learned binary codes.

Overall, we define the loss of entire matching stage as follows,

$$\begin{aligned}\mathcal{F}\left(\mathcal{A}|\mathbf{B}^s,\mathbf{B}^t\right) =&\mathcal{L}_{match} + \gamma_m \mathcal{L}_{hash} \\ &+ \eta_m \left(\|\vec{\phi}^s\|_F^2 + \|\vec{\phi}^t\|_F^2\right),\end{aligned} \tag{11}$$

where γ_m and η_m are hyper-parameters. The third term $\left(\|\vec{\phi}^s\|_F^2 + \|\vec{\phi}^t\|_F^2\right)$ regularizes the learned mapping results.

After obtaining the binary code of each user in two networks, we can use Hamming distance to calculate the similarity between binary codes and choose the most similar ones as the searching results.

4 OPTIMIZATION

In this section, we develop algorithm to estimate the parameters of the proposed model. We introduce optimization technics into two stages of learning hash codes, including negative sampling, pre-training and parameter sharing, achieving high computational efficiency and prediction accuracy.

4.1 Optimization on embedding stage

The direct optimization on Eq. (2) is computational expensive, requiring the summation over the entire set of users. To address this problem, we adopt the approach of negative sampling proposed in [15]. We define the negative sampling by the formalization as follows

$$\log\sigma\left({z'_j}^T \cdot z_i\right) + \sum_{k=1}^{K} E_{u_k\sim P_n(u)}\left[\log\sigma\left(-{z'_k}^T \cdot z_i\right)\right], \tag{12}$$

which is used to replace every $\log p\left(u_j|u_i\right)$ term in the objective function of the embedding stage. The function $\sigma(x) = 1/\left(1+\exp(-x)\right)$ is the sigmoid funciton. The negative sampling distinguishs the edge $\langle u_i, u_j\rangle$ from K noise edges $\langle u_i, u_j\rangle$ sampled from noise distribution $P_n(u)$. We set the noise distribution $P_n(u) \propto {d_u}^{3/4}$, where d_u is the out-degree of user u.

For further speeding up the optimization in embedding stage, we turn to optimize the embedding matrix $\mathbf{Z}^s$ and $\mathbf{Z}^t$ independently. For better explaining the optimization trick, we choose to illustrate the process by optimizing the embedding matrix $\mathbf{Z}^s$ at first[1]. After learning the embedding matrix $\mathbf{Z}^s$, we initialize the embedding matrix $\mathbf{Z}^t$ with the learned $\mathbf{Z}^s$ according to the labeled anchor links. Every j-th column of matrix $\mathbf{Z}^t$ are assigned by the user representation z_i^s if $(u_i^s, u_j^t) \in \mathcal{A}$. The other columns of matrix $\mathbf{Z}^t$ are assigned by random values and learned by optimizing the Eq. (4), while the preset columns of $\mathbf{Z}^t$ are still fixed if those have been assigned according to labeled anchor links. The pretrained parameters are used in optimizing the embedding matrix $\mathbf{Z}^t$. The parameters initialized by the pre-training method can be stabilized in training [11].

[1]The procedure can also be reverse. Alternatively, the embedding matrix $\mathbf{Z}^t$ can be learned firstly.

4.2 Optimization on matching stage

The optimization on logarithmic Eq. (8) also suffers high computational cost when summarizing compositions on all possible anchor links $\mathcal{A}$. To efficiently estimate the parameters on mapping, we maximize the $\log p\left(u_i^s, u_j^t\right)$ with negative sampling as

$$\log\sigma\left({\phi_j^t}^T \cdot \phi_i^s\right) + \sum_{k=1}^{K} E_{u_k\sim P'_n(u)}\left[\log\sigma\left({\phi_k^t}^T \cdot \phi_i^s\right)\right], \tag{13}$$

where the negative samples are drawn from $P'_n(u) = 1/|\mathcal{U}|$. Meanwhile, we construct two multilayer perceptrons as the mapping functions with shared parameters to handle the overfitting problem caused by scarce labeled anchor links in real applications. Except the independent parameters from the input layer to hidden layer, the parameters of the two mapping functions are shared with each other.

According to the formalization on matching stage, the best matching performance is achieved when the binary codes from labeled anchor links are set to be the same. Hence we regularize $b_{(i,j)} = b_i^s = b_j^t$, *iff* $(u_i, u_j) \in \mathcal{A}$ in the training process. If we only consider the observed anchor links, the function $\|\mathbf{B}^s - \vec{\phi}^s\|_F^2 + \|\mathbf{B}^t - \vec{\phi}^t\|_F^2$ can be unfolded as

$$\sum_{(u_i,u_j)\in\mathcal{A}}\left(\|b_{(i,j)}\|^2 + \|\vec{\phi}_i^s\|^2 + \|\vec{\phi}_j^t\|^2 - 2b_{(i,j)}^T \cdot \left(\vec{\phi}_i^s + \vec{\phi}_j^t\right)\right).$$

It can be easily derived that the minimization of function $\|\mathbf{B}^s - \vec{\phi}^s\|_F^2 + \|\mathbf{B}^t - \vec{\phi}^t\|_F^2$ is equivalent to maxmizing $b_{(i,j)}^T \cdot (\vec{\phi}_i^s + \vec{\phi}_j^t)$, when $\vec{\phi}^s$ and $\vec{\phi}^t$ are fixed. Therefore, we have $b_{(i,j)} = \mathrm{sign}(\vec{\phi}_i^s + \vec{\phi}_j^t)$.

At the end, we can reformulate the matching loss in Eq. (11) as follows,

$$\begin{aligned}\mathcal{F}'\left(\mathcal{A}|\mathbf{B}^s,\mathbf{B}^t\right) =& -\log\sigma\left({\phi_j^t}^T \cdot \phi_i^s\right) - \sum_{k=1}^{K}\left[\log\sigma\left({\phi_k^t}^T \cdot \phi_i^s\right)\right] \\ &+ \gamma_m \sum_{(u_i,u_j)\in\mathcal{A}}\left(\|b_{(i,j)} - \vec{\phi}_i^s\|^2 + \|b_{(i,j)} - \vec{\phi}_j^t\|^2\right) \\ &+ \eta_m\left(\|\vec{\phi}^s\|_F^2 + \|\vec{\phi}^t\|_F^2\right).\end{aligned}$$

The DHL-ALP alogorithm for parameter estimation is described in Algorithm 1.

5 EXPERIMENT

In experiments, we compare our proposed method to the state-of-the-art methods for anchor link prediction on both synthetic data and real data. The results show that the proposed method achieves the best performance in predicting anchor links with high prediction accuracy and time efficiency. Moreover, we conduct case studies on real data, presenting the high quality of anchor link candidate election in our proposed model.

Algorithm 1 DHL-ALP

```
Input: Social network $G^s$ and $G^t$, labeled anchor links $\mathcal{A}$.
Output: Network embedding $\mathbf{Z}^s$ and $\mathbf{Z}^t$, binary hash code matrices $\mathbf{B}^s$ and $\mathbf{B}^t$.

Initialization: Initialize parameters with random values, including $\mathbf{Z}^s$, $\mathbf{Z}^t$, $\phi^s$ and $\phi^t$.
1. Embedding stage
for $i = 1$ to $n$ do
    Randomly sample $m$ edges in network $G^s$ to construct a mini-batch.
    Calculate $\partial O(G^s)/\partial \mathbf{Z}^s$ and $\partial O(G^s)/\partial \mathbf{Z}'^s$.
    Update $\mathbf{Z}^s$ and $\mathbf{Z}'^s$ with stochastic gradient descent.
end for
Pretraining $\mathbf{Z}^t$ with the learned embedding matrix $\mathbf{Z}^s$ and labeled anchor links.
for $i = 1$ to $n$ do
    Randomly sample $m$ edges in network $G^t$ to construct a mini-batch.
    Calculate $\partial \left(O\left(G^t\right) + O\left(\mathcal{A}|G^s, G^t\right)\right)/\partial \mathbf{Z}^t$ and
    $\partial \left(O\left(G^t\right) + O\left(\mathcal{A}|G^s, G^t\right)\right)/\partial \mathbf{Z}'^t$ with fixed $\mathbf{Z}^s$.
    Update $\mathbf{Z}^t$ and $\mathbf{Z}'^t$ with stochastic gradient descent.
end for
2. Matching stage
for $i = 1$ to $n$ do
    Randomly sample $m$ anchor links from $\mathcal{A}$ to construct a mini-batch.
    Calculate $\partial \mathcal{F}'\left(\mathcal{A}|\mathbf{B}^s, \mathbf{B}^t\right)/\partial \phi^s$ and $\partial \mathcal{F}'\left(\mathcal{A}|\mathbf{B}^s, \mathbf{B}^t\right)/\partial \phi^t$.
    Update $\phi^s$ and $\phi^t$ with stochastic gradient descent.
end for
```

5.1 Comparative methods

Previous anchor link prediction methods seldom can be well applied to large-scale data, sufferring high computational cost in modeling and matching. To better illustrate the performance of our proposed model, we choose scalable methods as baselines. The compared methods are summarized as follows:

- **MNA (Multi-Network Anchoring)** [9]: This method models anchor link prediction as a classification issue. It extracts pairwise social features for an anchor link, including common neighbors, Jaccard coefficient and Adamic/Adar measure. The same features are adopted in our experiments.
- **PALE-MLP** [14]: This method is a two-stage model. After learning the user representations of two networks independently, it adopts a regression model to learn a mapping functions between user representations from source and target networks, implemented by multi-layer perceptron (MLP).
- **DHL-ALP (no cstr)**: To demonstrate the effectiveness of anchor-link-aware embedding, we add a variant of our proposed model as a baseline model. In embedding stage, DHL-ALP (no cstr) learns the embedding matrices of two networks independently without the constraint on labeled anchor links.
- **DHL-ALP**: The method is proposed in this paper. The difference between DHL-ALP (no cstr) and DHL-ALP is that DHL-ALP introduce the anchor-link-aware network embedding loss in embedding stage.

5.2 Experiments on synthetic data

The goal of experiments on synthetic data is to validate the effectiveness of our proposed models in anchor link prediction task, in which two main factors, i.e. the size of labeled anchor link set and the structural properties of two networks, influence the prediction accuracy. Thus, we conduct a series of experiments under different settings of labeled dataset size and network structural properties.

5.2.1 Dataset. The synthetic data was crawled from Blogcatalog[2], processed by Tang and Liu [24]. The dataset contains 10,312 bloggers and 333,983 social links of bloggers. We sample two sub-networks in Blogcatalog network in order to serve as source network and target network in anchor link prediction. All nodes in two sub-networks are inherited from the original network. The edges in each network are drawed from a piecewise function, parameterized by sparseness factor α_s and overlap factor α_c. We generate a random value p from a uniform distribution $p \sim U(0, 1)$ for each edge. If $p \in [0, 2\alpha_s + (1 - \alpha_s)\alpha_c - 1]$, the edge is discarded; If $p \in [2\alpha_s + (1 - \alpha_s)\alpha_c - 1, \alpha_s]$, the edge is kept in the first sub-network; If $p \in [\alpha_s, 1 - (1 - \alpha_s)\alpha_c]$, the edge is kept in the second sub-network; Otherwise, the edge is kept in both sub-networks. According to the sampling strategy, the larger α_s means the more discarded edges during sampling process, losing the diverse of network structure existing in original network. Meanwhile, the overlap factor α_c controls the structural similarity between two sub-networks. We assign one of sub-network as network G^s and another sub-network as network G^t. Every node in one sub-network has a matched node in the other sub-network. All matched node pairs are randomly splited into two datasets by a fixed proportion, denoted as training dataset and test datasets. Next, we conduct experiments with different settings on α_s, α_c and training data size in order to validate the effectiveness of our proposed model.

5.2.2 Evaluation and results analysis. We regard the prediction task of finding the matching node in another network as a ranking problem with the similarity score of two nodes' representations being served as ranking scores. Due to the high matching cost in baselines, 10 random nodes are sampled for each node. These sampled nodes, together with the ground truth matching node, are provided as candidate set. Each model is required to correctly recognize the ground truth matching node from the candidate set. The prediction performance is evaluate by *Mean Reciprocal Rank* (MRR). The larger MRR values indicate the better performance. To comphresensively evaluate our proposed model, we compare it with baselines in different settings. For each settings, we

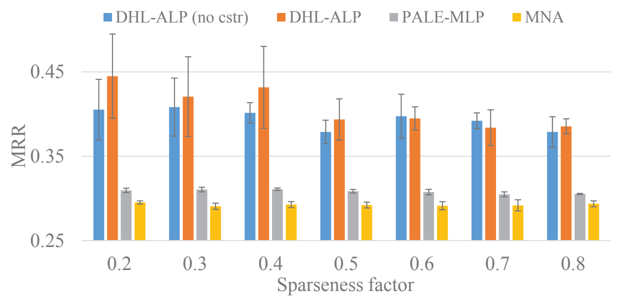
[2]http://www.blogcatalog.com

conduct 10 runs of experiments to examine the standard deviations of experimental results.

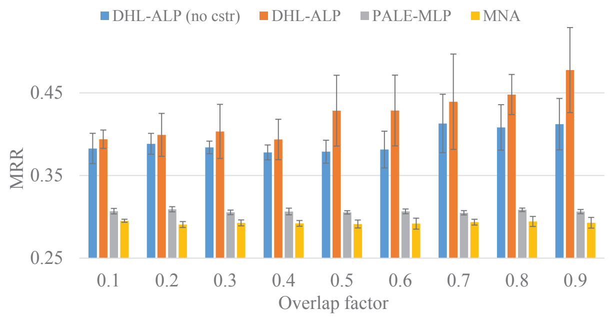
Firstly, we randomly pick up 60% of anchor links for training and choose $\alpha_s = \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$ with the same $\alpha_c = 0.8$ to sample sub-networks. The experimental results are shown in Fig. 2(a). As we can see in the figure, the MNA method achieves the worst performance on MRR. The prediction performance of PALE-MLP is much better than MNA. However, DHL-ALP and DHL-ALP (no cstr) perform consistently and significantly better than baselines on MRR. The prediction accuracy is decreasing with the increase of sparseness factor α_s for DHL-ALP and DHL-ALP (no cstr). According to the positive correlation between sparseness factor and diverse of network structure, the results indicate that the structural diversity kept in networks can help to improve the prediction accuarcy in anchor link prediction. It is interesting that the standard deviations of MRR on DHL-ALP and DHL-ALP (no cstr) are also decreasing with the increase of sparseness factor. Moreover, we can observe that DHL-ALP achieves better performance than DHL-ALP (no cstr) in most cases except for $\alpha = 0.6$ and 0.7. It means that the large sparseness factor value weakens the effects of anchor-link-aware embedding constraint in Eq. (6).

Then we use 60% of entire anchor links as traning data set and choose α_c from 0.1 to 0.9 with the same $\alpha_s = 0.5$ to sample sub-networks. The prediction results are presented in Fig. 2(b). As shown in the figure, DHL-ALP and DHL-ALP (no cstr) significantly outperform other baselines on MRR. In contrast to the prediction performance shown in Fig. 2(a), the prediction accuracy is increasing with the increase of overlap factor for DHL-ALP and DHL-ALP (no cstr). The results indicate that the overlap factor is positively correlated to the performance of anchor link prediction, and the larger structural similarity between two networks allows better accuracy. It can also be observed that DHL-ALP achieves better performance than DHL-ALP (no cstr) and the gap of performance is enlarged with the increase of overlap factor. It indicates that the network embedding with anchor-link-aware constraint is capable of learning homogeneous user representations when the two networks are similar to each other.

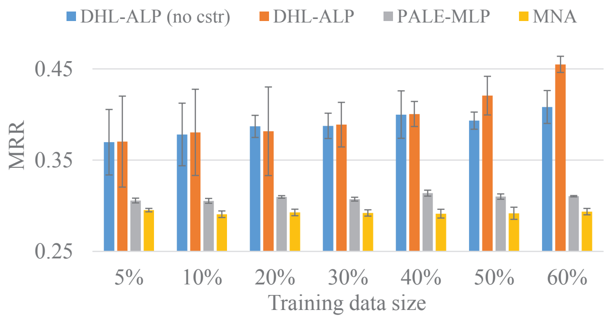
In addition, we also examine the effects of different training data size, varing the size from 5% to 60% of the entire labeled dataset. In this series of experiments, α_s and α_c are 0.8 and 0.3 respectively. The experimental results are shown in Fig. 2(c). In the figure, we can see that DHL-ALP and DHL-ALP (no cstr) still achieve the best performance on MRR under different training data size. The prediction accuracy is increasing with the increase of training data size for DHL-ALP and DHL-ALP (no cstr), while the standard deviations are reduced. It implies that the predictin performance of DHL-ALP and DHL-ALP (no cstr) is improved with the number of labeled anchor links. Meanwhile, it can be observed that DHL-ALP performs better than DHL-ALP (no cstr) in most situations, especially when there are plenty of training data.



(a) The performance on different sparseness factor α_s ($\alpha_c = 0.8$, the fraction of training data size is 60%)



(b) The performance on different overlap factor α_c ($\alpha_s = 0.5$, the fraction of training data size is 60%)



(c) The performance on different training data size ($\alpha_s = 0.8$, $\alpha_c = 0.3$)

Figure 2: The prediction performance on synthetic data with different settings.

In summary, according to the experimental results on synthetic data, we prove that our proposed models are significantly and consistently better than baselines in different settings on sparseness factor, overlap factor and training data size. Meanwhile, the performance gap between DHL-ALP and DHL-ALP (no cstr) indicate the effectiveness of the anchor-link-aware constraint in embedding stage.

5.3 Experiments on real data

The goal of the experiments on real data is to demonstrate the effectiveness of our proposed models in real applications.

The performance is evaluated by prediction accuracy, efficiency in retrieval time and the quality of anchor link candidate election.

5.3.1 Dataset. Networks from two popular social networking services in China are adopted as datasets:

- **Douban**[3]: Douban offers services for their users to freely share favorite content through social network, including movies, books, music and other off-line events.
- **Sina Weibo**[4]: Weibo is a very popular microblogging platform in China. The number of register users has exceeded 300 millions by the end of 2017.

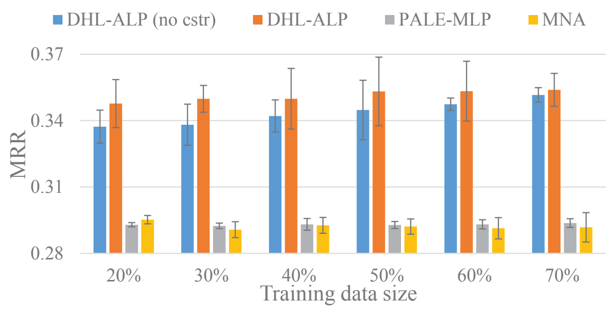
Table 1: Real data description

	#Users	#Links	#Anchor links
Douban	1,305,073	6,508,159	15,009
Weibo	324,716	4,428,026	

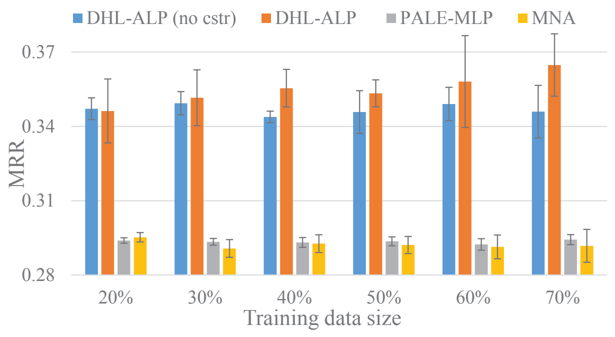
The anchor link set is crawled by API supported by the social networking services. In Douban, users would perfer to post the links to their own Weibo accounts in descriptions. According to the reliable information, we link 15,009 pairs of accounts across the two social networks. Then we extract the social networks based on the labeled users in each network. The basic statistics of extracted real data is illustrated in Table 1. As described in the table, we can see that only 4.6% of total Weibo users and 1.2% of total Douban users have their anchor links being labeled in our dataset. The scarce labeled anchor links with huge network size pose a big challenge in both model optimization and matching in anchor link prediction.

5.3.2 Evaluation on prediction accuracy. We conduct experiments on different training data size and examine whether the training data size has the same effects on prediction accuracy in real data. We choose the training data size from 20% to 70% and the corresponding experimental results are shown in Fig. 3(a) and 3(b). In the figure, "D $\rightarrow$ W" denotes the case where we try to find the linking Weibo account for a Douban account and vise versa for "W $\rightarrow$ D". We can observe that DHL-ALP (no cstr) and DHL-ALP exhibit the best performance on predicting anchor links in the prediction tasks of D $\rightarrow$ W and W $\rightarrow$ D and the prediction accuracy of DHL-ALP (no cstr) and DHL-ALP is increasing with the increase of training data size. With the 70% of entire labeled data for training, DHL-ALP achieves MRR values 0.3539 ± 0.0074 and 0.3647 ± 0.0126 in the prediction task D $\rightarrow$ W and W $\rightarrow$ D respectively, which improves the prediction results of PALE-MLP and MNA by over 20% in two tasks. Meanwhile, DHL-ALP outperforms DHL-ALP (no cstr) in all cases, staying consist with the observation in synthetic data. In the following experiments, we choose 70% of entire labeled anchor links as the training data.

[3]http://www.douban.com
[4]http://www.weibo.com



(a) D $\rightarrow$ W



(b) W $\rightarrow$ D

Figure 3: The prediction performance on real data with various training data size (the length of binary code is 20 bits).

Besides training data size, the length of binary code is also important in our proposed model when applying the proposed model in real applications. The length of binary code can be stored in bits, refering the capability of contained structural features, storage cost and retrieval speed. Therefore, we conduct experiments to check the prediction performance, varing the length of binary code in 16 bits, 20 bits, 24 bits, 28 bits and 32 bits respectively. We conduct 10 runs of experiments to retrieve the true anchor link for each node from a candidate set consisting of the ground truth mathcing node and 9 randomly sampled nodes, and record the experimental results in Table. 2. The prediction performance is evaluated by MRR and the best prediction results are marked in boldface in the table. As we can see in the table, the best prediction performance is achieved when the length of binary code is 20 bits. Short bits of binary codes may cause more conflicts in matching, resulting in the reduction of prediction performance. However, long bits of binary codes may suffer overfitting problem when the labeled anchor links is not enough to learn the matching model. Thus, we set the length of binary code to be 20 bits in our following experiments.

5.3.3 Time complexity. With the binary hash codes, we can retrieve the suspected anchor links by search methods,

Table 2: Prediction performance on different length of binary codes

Task	Method	16 bits	20 bits	24 bits	28 bits	32 bits
D $\rightarrow$ W	DHL-ALP (no cstr)	0.3554±0.0035	**0.3684±0.0140**	0.3433±0.0091	0.3354±0.0045	0.3330±0.0042
	DHL-ALP	0.3665±0.0074	**0.3893±0.0186**	0.3463±0.0202	0.3463±0.0078	0.3383±0.0110
W $\rightarrow$ D	DHL-ALP (no cstr)	0.3507±0.0103	**0.3538±0.0066**	0.3399±0.0070	0.3400±0.0073	0.3280±0.0094
	DHL-ALP	0.3628±0.0140	**0.3707±0.0171**	0.3508±0.0106	0.3372±0.0116	0.3400±0.0038

p.s. the best prediction performance of each model is marked in boldface.

Table 3: Time consuming when retrieving anchor links from Douban to Weibo (D $\rightarrow$ W) and from Weibo to Douban (W $\rightarrow$ D)

	Query number (Task: D $\rightarrow$ W)		
	1 node	100 nodes	1000 nodes
DHL-ALP	2.4s	235.1s	2531.3s
PALE-MLP	3.2s	316.6s	3230.4s
	Query number (Task: W $\rightarrow$ D)		
	1 node	100 nodes	1000 nodes
DHL-ALP	6.0s	618.6s	6095.3s
PALE-MLP	9.8s	1001.0s	9572.7s

such as linear scan, semantic hashing [19] and locality sensitive hashing [3]. Compared with semantic hashing and locality sensitive hashing, linear scan is the most straightforward and precise method for retrieving the best candidates, which computes the distance between the query user representation and every possible user representation in the datasets. However, most previous works can only adopt linear scan strategy for predicting the anchor links. For directly demonstrating the effectiveness of our proposed model in time complexity of matching, we also choose linear scan strategy applied in our proposed model. We choose PALE-MLP as the competitive method, since PALE-MLP adopt a simple distance calculation algorithm in matching. The experiments are implemented with Python 2.7 on Intel(R) Xeon(R) CPU ES-2640 @ 2.40GHz with 10 cores and 128GB memory. The experimental results are shown in Table. 3. We can see that DHL-ALP spend less time in query anchor links for 1 node, 100 nodes and 1000 nodes. In the task D $\rightarrow$ W, DHL-ALP spend approximate 2.4s for one single query, 235.1s for 100 queries and 2531.3s for 1,000 queries. The time comsuming is less than PALE-MLP by nearly 25%. In the task W $\rightarrow$ D, the time consuming of DHL-ALP is less than PALE-MLP by nearly 38%. Moreover, the gap of retrieval time between DHL-ALP and PALE-MLP would be enlarged by the number of dataset size.

5.3.4 Case study. Although the binary hash codes improve the retrieval speed in matching, the hash codes may cause great number of conflicts causing negative effects indeed. Therefore, we randomly pick up 5 query instances in the task D $\rightarrow$ W and examine the number of recall and conflicts in case studies. The query is anonymized by only preserving two head and tail characters respectively. The experimental results are shown in Table. 4. As shown in the table, we can observe that all ground truth of anchor links related to queries is located in the result set where the filtering condition (Hamming distance–HD) is equal to 0. Meanwhile, the number of conflicts of retrieval results is little in comparsion to the dataset size. For example, there are only 14 results in the query “De*dx” under the filtering condition HD$\leq$0, which filters out 99.996% of entire users in Weibo dataset. It indicates that our proposed method is efficient with high recall and low conflicts in real applications.

Table 4: Case study: the number of conflicts when retrieving by binary hash codes from DHL-ALP.

Task: D $\rightarrow$ W	Threshold t (HD $\leq t$)				include true
Douban account	0	1	2	3	answers?
ki*sn	**75**	1295	3475	11180	Yes
sh*ir	**52**	118	475	1628	Yes
lu*a5	**77**	206	1539	3850	Yes
De*dx	**14**	90	797	1256	Yes
Mr*AA	**17**	56	220	622	Yes

p.s. HD refers to Hamming distance. The numbers with boldface mean that the true answers are located in the corresponding set.

6 CONCLUSION

In this paper, we present a big challenge for anchor link prediction, that is, existing methods can hardly be deployed in practical system as the high computation cost when identifying anchor links in large-scale social networks. To combat this problem, we propose a novel deep learning architecture for learning binary hash codes across networks. The binary hash code has characteristics in low storage cost and fast retrieval speed so as to fast index candidate users in anchor link prediction. The proposed method is implemented in two anchor-link-aware stages, named network embedding and matching. In the stage of network embedding, we learn embedding matrix and embed each user from high-dimensional and sparse raw representaion into a low-dimensional and dense representaion in each network. Furthermore, we propose an anchor-link-aware embedding constraint to regularize the user representations with labeled anchor links. In the matching stage, we propose an efficient matching model based on learned user representations. The matching model export binary hash codes without much loss of similarity preserved in original user representations.

We evaluate the effectivenss of our proposed models on both synthetic dataset and large-scale real dataset. Experimental results demonstrate that our proposed models can consistently and significantly outperform state-of-the-art modeling methods at both prediction accuracy and time efficiency in anchor link prediction. Additionally, DHL-ALP performs better than DHL-ALP (no cstr) in most cases, implying the effectiveness of anchor-link-aware constraint in network embedding. We also analyze the time complexity and conduct case studies for examining the number of conflicts and recall on our proposed model. The experimental results show the great potential of our proposed model in real applications.

REFERENCES

[1] Yi Cui, Jian Pei, Guanting Tang, Wo Shun Luk, Daxin Jiang, and Ming Hua. 2013. Finding email correspondents in online social networks. *World Wide Web* 16, 2 (2013), 195–218. https://doi.org/10.1007/s11280-012-0168-2

[2] Ruben Enikolopov, Maria Petrova, and Konstantin Sonin. 2018. Social media and corruption. *American Economic Journal: Applied Economics* 10, 1 (2018), 150–74.

[3] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. 1999. Similarity search in high dimensions via hashing. In *Vldb*, Vol. 99. 518–529.

[4] Oana Goga, Patrick Loiseau, Robin Sommer, Renata Teixeira, and Krishna P. Gummadi. 2015. On the Reliability of Profile Matching Across Large Online Social Networks. (2015). https://doi.org/10.1145/2783258.2788601 arXiv:1506.02289

[5] Qing-Yuan Jiang and Wu-Jun Li. 2016. Deep Cross-Modal Hashing. (2016), 3232–3240. https://doi.org/10.1109/CVPR.2017.348 arXiv:1602.02255

[6] Qing-Yuan Jiang and Wu-Jun Li. 2017. Asymmetric Deep Supervised Hashing. (2017). arXiv:1707.08325 http://arxiv.org/abs/1707.08325

[7] Gunnar W. Klau. 2009. A new graph-based method for pairwise global network alignment. *BMC Bioinformatics* 10, SUPPL. 1 (2009). https://doi.org/10.1186/1471-2105-10-S1-S59

[8] Giorgos Kollias, Shahin Mohammadi, and Ananth Grama. 2011. Network Similarity Decomposition (NSD): A Fast and Scalable Approach to Network Network Similarity Decomposition (NSD): A Fast and Scalable Approach to Network Alignment. 24, January (2011), 2232–2243.

[9] Xiangnan Kong, Jiawei Zhang, and Philip S. Yu. 2013. Inferring anchor links across multiple heterogeneous social networks. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management - CIKM '13*. 179–188. https://doi.org/10.1145/2505515.2505531

[10] Mehmet Koyutürk, Yohan Kim, Umut Topkara, Shankar Subramaniam, Wojciech Szpankowski, and Ananth Grama. 2006. Pairwise Alignment of Protein Interaction Networks. *Journal of Computational Biology* 13, 2 (2006), 182–199. https://doi.org/10.1089/cmb.2006.13.182

[11] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521 (2015), 436–444.

[12] Chung-Yi Li and Shou-De Lin. 2014. Matching users and items across domains to improve the recommendation quality. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*. 801–810. https://doi.org/10.1145/2623330.2623657

[13] Siyuan Liu, Shuhui Wang, Feida Zhu, Jinbo Zhang, and Ramayya Krishnan. 2014. HYDRA: Large-scale Social Identity Linkage via Heterogeneous Behavior Modeling. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14)*. ACM, New York, NY, USA, 51–62. https://doi.org/10.1145/2588555.2588559

[14] Tong Man, Huawei Shen, Shenghua Liu, Xiaolong Jin, and Xueqi Cheng. 2016. Predict anchor links across social networks via an embedding approach. *IJCAI International Joint Conference on Artificial Intelligence* 2016-Janua (2016), 1823–1829.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*. 3111–3119.

[16] Xin Mu, Feida Zhu, Ee-Peng Lim, Jing Xiao, Jianzong Wang, and Zhi-Hua Zhou. 2016. User identity linkage by latent user space modelling. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1775–1784.

[17] Arvind Narayanan and Vitaly Shmatikov. 2010. Myths and fallacies of personally identifiable information. *Commun. ACM* 53, 6 (2010), 24–26.

[18] Yuanping Nie, Yan Jia, Shudong Li, Xiang Zhu, Aiping Li, and Bin Zhou. 2016. Identifying users across social networks based on dynamic core interests. *Neurocomputing* 210 (2016), 107–115. https://doi.org/10.1016/j.neucom.2015.10.147

[19] Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning* 50, 7 (2009), 969–978.

[20] C Shi, Y Li, J Zhang, Y Sun, and P S Yu. 2017. A Survey of Heterogeneous Information Network Analysis. *IEEE Transactions on Knowledge and Data Engineering* 29, 1 (2017), 17–37. https://doi.org/10.1109/TKDE.2016.2598561

[21] Kai Shu, Suhang Wang, Jiliang Tang, Reza Zafarani, and Huan Liu. 2017. User Identity Linkage Across Online Social Networks: A Review. *SIGKDD Explor. Newsl.* 18, 2 (2017), 5–17. https://doi.org/10.1145/3068777.3068781

[22] Rohit Singh, Jinbo Xu, and Bonnie Berger. 2007. Pairwise global alignment of protein interaction networks by matching neighborhood topology. In *Research in computational molecular biology*. Springer, 16–31.

[23] Jiliang Tang, Yi Chang, and Huan Liu. 2014. Mining social media with social theories: a survey. *ACM SIGKDD Explorations Newsletter* 15, 2 (2014), 20–29.

[24] Lei Tang and Huan Liu. 2009. Relational learning via latent social dimensions. *Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining* (2009), 817–825. https://doi.org/10.1145/1557019.1557109

[25] Daixin Wang, Peng Cui, Mingdong Ou, and Wenwu Zhu. 2015. Deep multimodal hashing with orthogonal regularization. *IJCAI International Joint Conference on Artificial Intelligence* 2015-Janua, Ijcai (2015), 2291–2297.

[26] Dan Wang, Heyan Huang, Chi Lu, Bo-Si Feng, Liqiang Nie, Guihua Wen, and Xian-Ling Mao. 2017. Supervised Deep Hashing for Hierarchical Labeled Data. (2017). arXiv:1704.02088 http://arxiv.org/abs/1704.02088

[27] Yongqing Wang, Huawei Shen, Shenghua Liu, Jinhua Gao, and Xueqi Cheng. 2017. Cascade dynamics modeling with attention-based recurrent neural network. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 2985–2991.

[28] Reza Zafarani and Huan Liu. 2009. Connecting Corresponding Identities across Communities. *Proceedings of the Third International Conference on Weblogs and Social Media - ICWSM 2009* 9, November (2009), 354–357. http://www.aaai.org/ocs/index.php/ICWSM/09/paper/viewFile/209/538

[29] Reza Zafarani and Huan Liu. 2014. Users Joining Multiple Sites : Distributions and Patterns User Membership Distribution across Sites. *Aaai* (2014), 635–638.

[30] Reza Zafarani and Huan Liu. 2016. Users joining multiple sites: Friendship and popularity variations across sites. *Information Fusion* 28 (2016), 83–89.

[31] Reza Zafarani, Lei Tang, and Huan Liu. 2015. User Identification Across Social Media. *Tkdd* 10, 2 (2015), 16. https://doi.org/10.1145/2747880

[32] Jiawei Zhang, Jianhui Chen, Shi Zhi, Yi Chang, S Yu Philip, and Jiawei Han. 2017. Link prediction across aligned networks with sparse and low rank matrix estimation. In *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*. IEEE, 971–982.

[33] Jiawei Zhang and Philip S. Yu. 2015. Integrated anchor and social link predictions across social networks. *IJCAI International Joint Conference on Artificial Intelligence* 2015-Janua, Ijcai (2015), 2125–2132.

[34] Xiaoping Zhou, Xun Liang, Xiaoyong Du, and Jichao Zhao. 2017. Structure Based User Identification across Social Networks. *IEEE Transactions on Knowledge and Data Engineering* (2017).