Disentangled Network Alignment with Matching Explainability

Fan Zhou*, Zijing Wen*, Goce Trajcevski‡, Kunpeng Zhang§, Ting Zhong*†, Fang Liu*,
*School of Information and Software Engineering, University of Electronic Science and Technology of China
‡Department of Electrical and Computer Engineering, Iowa State University, Ames IA
§Robert H. Smith School of Business, University of Maryland, College park MD
†Corresponding author: zhongting@uestc.edu.cn

***Abstract*—Network alignment (NA) is a fundamental problem in many application domains – from social networks, through biology and communications, to neuroscience. The main objective is to identify common nodes and most similar connections across multiple networks (resp. graphs). Many of the existing efforts focus on efficient anchor node linkage by leveraging various features and optimizing network mapping functions with the pairwise similarity between anchor nodes. Despite the recent advances, there still exist two kinds of challenges: (1) entangled node embeddings, arising from the *contradictory* goals of NA: embedding proximal nodes in a closed form for representation in a single network vs. discriminating among them when mapping the nodes across networks; and (2) lack of interpretability about the node matching and alignment, essential for understanding prediction tasks. We propose *d*NAME (*disentangled* Network Alignment with Matching Explainability) – a novel solution for NA in heterogeneous networks settings, based on a matching technique that embeds nodes in a disentangled and faithful manner. The NA task is cast as an adversarial optimization problem which learns a proximity-preserving model locally around the anchor nodes, while still being discriminative. We also introduce a method to explain our semi-supervised model with the theory of robust statistics, by tracing the importance of each anchor node and its explanations on the NA performance. This is extensible to many other NA methods, as it provides model interpretability. Experiments conducted on several public datasets show that *d*NAME outperforms the state-of-the-art methods in terms of both network alignment precision and node matching ranking.**

***Index Terms*—network alignment, social networks, graph kernel, adversarial learning**

I. Introduction

Network Alignment (NA) targets the identification of vertices across different social platforms (or other networks like, e.g., protein networks [1]) that refer to the same individual/entity [2]. It has recently attracted an increased amount of interest due to its crucial impact in various applications settings such as user behavior prediction [3], and identity verification and privacy protection [4]. Variants of the NA problem are also known as User Identity Linkage [5], [6], Account Linkage Inference [7], [8] and Anchor Link Prediction [9]–[11].

Various approaches have been proposed to tackle this practically relevant problem – e.g., COSNET [12] computes Adamic/Adar scores to measure neighborhood similarities via capturing the local and global consistency among multiple networks; FINAL [13] aligns attributed networks by leveraging the node/edge attribute information to guide topology-based alignment process; etc. The commonality of all these methods is that they extract a set of independent features from account profiles or activities (e.g., username, gender, writing style, etc.) for representing users – which may not be feasible due to various reasons, such as privacy and data availability. Inspired by recent advances in graph representation [14], many approaches have been proposed to address the NA problem using network embedding techniques. For example: – IONE [15] learns network embedding from the follower/followee couplings in order to preserve the proximity of users with "similar" relationships. –ULink [5] builds the latent user space through projection matrix, and jointly optimizes the matching pair information across different networks. – SLAMPRED [11] projects the features extracted for links from different aligned networks into a shared lower-dimensional feature space and predicts the anchor link according to link similarity. – DeepLink [6] addresses the network alignment problem with a semi-supervised dual learning framework where network embedding and neural network based mapping are employed for representing and linking anchor nodes.

Challenges and Contributions: Although existing NA methods are effective; with various carefully designed cross-network embeddings – they suffer two main drawbacks:
(1) Two *contradictory objectives* exist in these NA methods. That is, for a single network, it is desirable to embed proximity nodes in closed forms (or low-dimensional vectors), which has been done by most (if not all) graph representation algorithms, such as DeepWalk [16] and node2vec [17] among many others [14]. On the contrary, when learning the mapping/aligning functions between heterogeneous networks, it is desirable to explicitly separate the embedding vector of each node from the vectors of its proximities; otherwise, it is difficult to discriminate the anchor nodes from their neighborhoods, resulting in poor alignment performance. These two competing goals explain why existing alignment algorithms demonstrate high *top-k* ($k > 1$) linking accuracy but hardly improve the *top-1* accuracy especially for dense networks – which we refer to as the confounding matching problem. Since the representation of nodes in each network is entangled locally and even more so after the mapping, enhancing the graph

embedding resolution is the main challenge in solving the network alignment problem.
(2) Although graph embedding based approaches provide the state-of-the-art alignment performance, these end-to-end methods lack *explanability* regarding the process of fine-grained node matching and alignment. For example, why does a particular embedding and mapping work for a particular NA task? What is the glass-ceiling of these approaches and how to measure the bounds? Which samples are positive for successful linkage and which of them have negative influence? If we can distinguish positively influential nodes, then how much is the impact of each of them on predicting the anchor nodes? Answering such questions is not trivial, and yet they are fundamental for understanding the NA task, which can benefit the model development – e.g., decreasing weights of those identified negative samples may help improve the aligning performance.

To address the above limitations, we present *d*NAME (*disentangled* Network Alignment with Matching Explainability) – a novel approach reconciling the competing goals of existing NA methods, while capable of explaining the influence of each node on the network matching. Methodologically, *d*NAME exploits graph convolutional neural networks (GCN) for learning the latent space of single network combined with a graph kernel based regularizer for discriminating the representation of anchor nodes from their neighborhoods. Adversarial learning paradigm is employed to further distinguish the anchor nodes from the non-anchor but nearby ones. In addition, we leverage the influence functions [18], [19] to explain the network alignment performance by identifying the influential anchor nodes (both positive and negative) for aligning two networks. Our main contributions are:

• **Discriminative Convolutions**: We develop a regularized graph convolution to learn latent semantics of both node representation and network structure, while a graph kernel is employed for discriminating the anchor nodes from their neighborhoods. In addition to efficiently leveraging the node labels in a semi-supervised manner and the ability of being generalized to various graphs (besides social networks), *d*NAME bridges the tight clustered embedding in NA and abundant graph kernels originally used in graph classification.

• **Adversarial Matching**: We leverage upon generative neural networks to facilitate the confounding matching issue via a minimax game between a generative mapping function and a matching discriminator, which are jointly optimized to disentangle the matching results by preserving the relative similarity ordering and stabilize the alignment process.

• **Explainable Alignment**: *d*NAME is capable of explaining the behavior of node linkage by investigating the perturbation of each training sample and its influence on node alignment. This could benefit the potential network alignment algorithms with the interpretability of their models.

• To demonstrate the effectiveness of *d*NAME, we conducted experiments on several real-world datasets. The results show that *d*NAME can both improve the network alignment accuracy compared to the state-of-the-art approaches, and also explain its behavior.

In the rest of this paper, Section II presents the preliminary background and formalizes the problem. We provide model-free evidence in Section III, motivating the methodology including disentangled embedding and adversarial matching in Section IV, followed by the model interpretability in Section V. Experimental evaluations quantifying the benefits of our approach are presented in Section VI, and Section VII concludes the paper and outlines directions for future work.

II. Preliminary & Related Work

We now introduce the basic terminology and definitions in *d*NAME setting, and survey the related works.

A. Problem Definition

We consider a set of s different social networks $\{\mathcal{G}^1, ..., \mathcal{G}^s\}$, each defined as a Social Network Graph (SNG) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, unweighted and undirected. $\mathcal{V}$ is the set of vertices – each representing a user; and $\mathcal{E}$ is the set of edges – $e_{i,j} \in E$ indicating a relationship between the users u_i and u_j.

We represent each SNG with a unique low-dimensional latent user space according to the Network Embedding (NE) model, which learns the probabilistic distributions of nodes while preserving network properties. After embedding each SNG (obtaining $\mathbf{G}^1, ..., \mathbf{G}^n$, and we also use boldface to denote embedded nodes, e.g., $\mathbf{u}_i$), approximate graph mapping (projection) functions are developed as the task of NA [5], [6], [15], [20]. We note that some implementations consider node attributes [5], [20] while others do not [6], [15].

Definition 1. *(**Network Mapping Function**) A function Φ is a mapping from $\mathcal{G}^s$ to $\mathcal{G}^t$, if for each $u_i \in \mathcal{G}^s$ and its latent space vector $\mathbf{u}_i$, we have $\Phi(\mathbf{u}_i) = \mathbf{u}'_i$, where $\mathbf{u}'_i$ is in the latent space of $\mathcal{G}^t$.*

In general, the mapping function Φ is unknown – and the objective of NA methods is to learn a mapping Φ such that two SNGs $\mathcal{G}^s$ and $\mathcal{G}^t$ are aligned by maximizing the similarity of all aligned pairs $(\mathbf{u}_i, \mathbf{v}_j)$, where $\mathbf{u}_i \in \mathbf{G}^s$ and $\mathbf{v}_j \in \mathbf{G}^t$.

Over the last decade, graph kernels [21] have become the most effective approach for this task, providing a way of applying general kernel methods to graphs. Formally:

Definition 2. *A **Graph Kernel** is a positive semidefinite kernel function $\kappa : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ such that there exists a map $\phi : \mathbf{G} \to \mathbb{F}$ into a Hilbert space $\mathbb{F}$, with the property $\kappa(\mathbf{G}^i, \mathbf{G}^j) = \langle \phi(\mathbf{G}^i), \phi(\mathbf{G}^j) \rangle_{\mathbb{F}}$ for any pair of graphs $\mathcal{G}^i$ and $\mathcal{G}^j$, where $\langle \cdot, \cdot \rangle$ is the inner product in $\mathbb{F}$.*

Graph kernels have traditionally been employed as efficient functions to measure the similarity of a pair of (sub)graphs. There exist a plethora of graph kernels – e.g., shortest paths, random walks and subgraphs, (cf. [22] for a review), and considerable improvements are possible by using the Weisfeiler-Lehman test of isomorphism [23].

B. Related Work

Network-based methods have received much attention and become increasingly promising in tackling the NA problem, when only network structure is exploited for alignment, based on a few known anchor nodes. In [24], Tan et al. model user relationship using a hyper-graph and project the manifolds of two networks onto a commonly embedded space to correlate accounts. Neighborhood-based features seem like a natural choice for the user-identity linkage problem [5], [12], [25], relying on computing the Adamic/Adar scores to measure the neighborhood similarity [26]. CLF [27] predicts correlation for both anchor nodes and social links by transferring information related to social links formed by anchor nodes in the source network to the target network.

Inspired by word2vec [28] techniques in natural language processing, a number of approaches have been proposed to represent the graph with low-dimensional vectors, e.g., DeepWalk [16], node2vec [17] and GCN [29] – c.f [14] for an overview. Recently, some researchers have exploited network embedding methods for solving the network alignment problem, i.e., DeepLink [6] among others [5], [11]. However, the entangled embedding and indistinguishable matching problems have not been well studied in this area.

Compared to these works, *d*NAME is capable of discriminating the anchor nodes from their neighborhoods. In addition, it learns node matching in an adversarial manner, which makes the distribution of anchor nodes widely spread and yields higher representation resolution for improving alignment accuracy. To explain the aligning behavior of *d*NAME, we borrow the idea of influence functions [18], [19] to investigate the training process while making our methods robust to model interpretability.

III. Model-Free Evidence

In this section, we present model-free evidence to assess the impact of various factors on NA and to motivate our model.

A. Graphs Similarity

As mentioned, the existing NA methods focus on leveraging node features and/or employing efficient network representation (local and global) and mapping functions. However, to align a pair of networks, we need to understand the notion of their similarity, since this implicitly determines the upper-bound performance of a specific NA method. Furthermore, understanding the glass-ceiling of the datasets may, in turn, help incubating more efficient NA approaches.

We propose to measure the graph similarity with Optimal Transport (OT) cost – i.e., Earth Mover's Distance (EMD) – originally used for measuring divergences between two probability distributions [30], and recently employed as a way of designing graph kernels [31], [32].

Given two networks $\mathcal{G}^s = (\mathcal{V}^s, \mathcal{E}^s)$ and $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t)$, we embed each network with a matrix as a disentangled GCN model, where each row of the matrix represents the embedding of a node. This enables a formalization of the similarity comparison of two graphs as a transportation problem solved via EMD [30]. In the case of measuring network similarity, we define the OT cost between a pair of networks $(\mathcal{G}^s, \mathcal{G}^t)$ as:

$$\min \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} \mathbf{T}_{ij} d(\mathbf{u}_i, \mathbf{v}_j) \tag{1}$$

$$\text{s.t.} \sum_{i=1}^{n_1} \mathbf{T}_{ij} = \frac{1}{n_2}, \sum_{j=1}^{n_2} \mathbf{T}_{ij} = \frac{1}{n_1}, \mathbf{T}_{ij} \geq 0, \quad \forall i \in \{1, ..., n_1\}, \forall j \in \{1, ..., n_2\} \tag{2}$$

(cf. [32]), where $d(\mathbf{u}_i, \mathbf{v}_j)$ is a measure of vertex dissimilarity [33] between nodes $u_i \in \mathcal{V}^s$ and $v_j \in \mathcal{V}^t$, and $\mathbf{T} \in \mathcal{R}^{n_1 \times n_2}$ ($n_1 = |\mathcal{V}^s|$ and $n_2 = |\mathcal{V}^t|$) is a transportation matrix. The node dissimilarity $d(\mathbf{u}_i, \mathbf{v}_j)$ between u_i and v_j is computed as $d(\mathbf{u}_i, \mathbf{v}_j) = (1 - \cos\langle \mathbf{u}_i, \mathbf{v}_j \rangle)/2$, where $\cos\langle \mathbf{u}_i, \mathbf{v}_j \rangle$ is the cosine similarity between nodes u_i and v_j. Note that $d(\mathbf{u}_i, \mathbf{v}_j) = 0$ means that u_i and v_j are exactly the same in the latent representation.

Regarding the transportation matrix $\mathbf{T}$, its element $\mathbf{T}_{ij} \geq 0$ denotes how much mass from the vertex $u_i \in \mathcal{V}^s$ "travels" to the vertex $v_j \in \mathcal{V}^t$. This formulation allows each node $u_i \in \mathcal{V}^s$ to be transported into any node $v_j \in \mathcal{V}^t$ in total or in parts [31]. We note that the outgoing mass from each graph should be equal to 1 and is equally divided among all the vertices. That is, we want to ensure that the entire outgoing mass from vertex u_i amounts to $\frac{1}{n_1}$ – i.e., $\sum_j \mathbf{T}_{ij} = \frac{1}{n_1}$. In addition, the amount of incoming mass to vertex v_j should match $\frac{1}{n_2}$, i.e., $\sum_i \mathbf{T}_{ij} = \frac{1}{n_2}$. The OT distance between the two networks is defined as the minimum cumulative cost required to move all the nodes from $\mathcal{G}^s$ to $\mathcal{G}^t$.

However, the above method does not distinguish anchor nodes from non-anchor ones. In the context of NA, the anchor nodes $u_i \in \mathcal{V}^s$ can only be transported to the anchor nodes $v_j \in \mathcal{V}^t$. Thus, we set the distance between anchor nodes to be $d(\mathbf{u}_i, \mathbf{v}_j)$ as above, and the distance between non-anchor nodes to be maximal – which is 1, according to the range of $d(\mathbf{u}_i, \mathbf{v}_j)$. Hence, we normalize and recalculate the dissimilarity distance as:

$$d'(\mathbf{u}_i, \mathbf{v}_j) = \begin{cases} d(\mathbf{u}_i, \mathbf{v}_j), & \text{if } u_i \text{ and } v_j \text{ are aligned anchor nodes;} \\ 1, & \text{otherwise.} \end{cases} \tag{3}$$

Fig. 1 illustrates the performance of NA between two networks by manually varying the graph similarity measured using EMD. Apparently, the node alignment accuracy deteriorates with the increasing dissimilarity of two graphs.

B. Indistinguishable Embedding & Matching

There are two indistinguishable situations where the aligning performance may deteriorate:

(1) In the network embedding stage, it is not easy to distinguish a node from its proximal ones under existing unsupervised graph representation techniques. For example, Fig. 2 shows the latent representation (after further dimension reduction via t-SNE) of a graph embedding (Twitter [34], Table I) using deepWalk and node2vec, respectively. We can observe

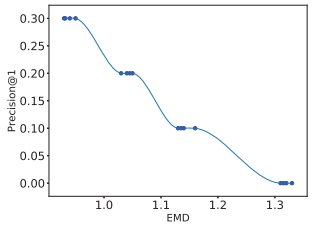
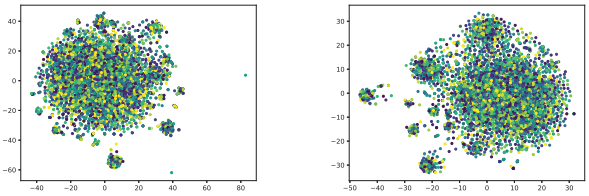


Fig. 1. Performance of Network Alignment vs. Graph Similarity. Two networks: MySpace and Lastfm [12] (cf. Table I in Sec. VI). Network Embedding: node2vec [17]; Mapping function: MLP [6].



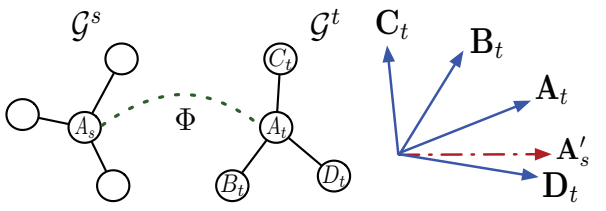
(a) deepWalk. (b) node2vec.

Fig. 2. Indistinguishable Representation via Network Embedding. We note that t-SNE is applied to obtain a 2-dimensional vector representation of each node for visualization.

that the nodes (dots in the Figure) are indistinguishable with each other.

(2) The confounding matching problem with learned mapping function occurs when estimating the similarity between nodes. For example, the MultiLayer Perceptron (MLP) are usually employed to learn the mapping function between two graphs [5], [6], where the objective is to minimize the overall loss between anchor node pairs. However, one side effect is that we cannot efficiently discriminate the real anchor nodes with their neighborhoods, since the point-wise ranking of vectors in the target network is perturbed by the network embedding process, and the confounding matching may be further deteriorated after the mapping, as illustrated in Fig. 3.

These two issues are crucial for improving the NA performance and motivate our investigation of discriminative



(a) Mapping between two networks. (b) Confounding matching.

Fig. 3. Confounding matching: After learning the mapping function Φ between two networks (part 3(a)), node A_s is mapped to $\mathcal{G}^t$ with $\Phi(\mathbf{A}_s) = \mathbf{A}'_s$. However, in the latent space of $\mathcal{G}^t$ (part 3(b)), $\mathbf{A}'s$ is closer towards $\mathbf{D}_t$ than to $\mathbf{A}_t$ (the true corresponding node of A in the target network $\mathcal{G}^t$) – incurring a linking error due to tightly clustered embedding space of proximal nodes not allowing enough resolution to distinguish them, especially for the mapped vectors like $\mathbf{A}'s$.

embedding and confounding matching.

IV. Disentangled Network Alignment

We now present our method for representing nodes with a disentangled graph convolution model and matching them across graphs in an adversarial manner.

A. Discriminative Graph Convolutional Embedding

Representing each node in a network and capturing the latent network structural semantics can be accomplished by various network embedding (NE) based methods. In the context of *d*NAME, we are more interested in neighboring nodes than distant ones for linking a user u_i across networks, because the features and connectivity of nearby nodes provide more useful information or additional context. That is, *local* structure information has more impact in predicting node relations than higher-order proximity of the network.

Therefore, we propose a disentangled Graph Convolutional Network (GCN) embedding method, focusing on iteratively aggregating feature information from local graph neighborhoods. This is an efficient approach – learning polynomials of the graph Laplacian avoids the computation of eigenvectors [35]. We also leverage the semi-supervised graph learning ability of GCN to discriminate anchor nodes from non-anchor nodes.

Graph Convolution: we represent each $\mathcal{G}^i$ with two matrices: a symmetric adjacency matrix $\mathbf{W} \in \mathbb{R}^{N\times N}$ ($W_{ij} = 0$ if $(i,j) \notin \mathcal{E}$ and $W_{ij} > 0$ otherwise) and a diagonal degree matrix $\mathbf{D} \in \mathbb{R}^{N\times N}$ ($D_{ii} = \sum_j W_{ij}$). They can be obtained through graph Laplacian operations. The (unnormalized) graph Laplacian is a symmetric positive semidefinite matrix $\mathbf{L}_u = \mathbf{D} - \mathbf{W}$. By a symmetric normalization, one can obtain a normalized graph Laplacian $\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{I}$ is the identity matrix. Since $\mathbf{L}$ is also symmetric and positive-semidefinite, it can be diagonalized as $\mathbf{L} = \mathbf{\Psi}\mathbf{\Lambda}\mathbf{\Psi}^\intercal$, where $\mathbf{\Lambda} = \text{diag}(\lambda_1, \cdots, \lambda_N)$ are the spectrum (non-negative eigenvalues) of $\mathbf{L}$ and $\mathbf{\Psi} = (\psi_1, \cdots, \psi_N)$ are the corresponding orthonormal eigenvectors.

Representation Learning: Generalizing convolution and F filters to a signal $\mathbf{X} \in \mathbb{R}^{N\times d}$ – where each row (e.g., a node/user) is a d-dimensional feature vector $\mathbf{x}_i$ – produce a convolved signal matrix $\mathbf{U} \in \mathbb{R}^{N\times F}$

$$\mathbf{U} = \widehat{\mathbf{D}}^{-\frac{1}{2}}\widehat{\mathbf{W}}\widehat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{F} \tag{4}$$

where $\mathbf{F} \in \mathbb{R}^{d\times F}$ is a matrix of filter parameters, $\widehat{\mathbf{W}} = \mathbf{W} + \mathbf{I}$ is the re-parameterization trick and the diagonal elements of matrix $\widehat{\mathbf{D}}$ are $\widehat{D}_{ii} = \sum_j \widehat{W}_{ij}$.

Suppose we have an L-layer neural network. It can be trained with layer-wise propagation rule from input to output with the following differentiable function and trainable parameters in each layer l:

$$\begin{aligned}\mathbf{H}^l &= \sigma(\widehat{\mathbf{D}}^{-\frac{1}{2}}\widehat{\mathbf{W}}\widehat{\mathbf{D}}^{-\frac{1}{2}}\sigma(\mathbf{U}^{l-1})\mathbf{F}^{l-1}) \\ &= \sigma(\widehat{\mathbf{D}}^{-\frac{1}{2}}\widehat{\mathbf{W}}\widehat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{l-1}\mathbf{F}^{l-1})\end{aligned}$$

where $\mathbf{U}^{l-1}$ is the convolved signal matrix in the $(l-1)^{th}$ layer; $\mathbf{F}^{l-1} \in \mathbb{R}^{F^{l-1}\times F^l}$ is a layer-specific trainable weight

matrix; $\sigma(\cdot)$ is an activation function (e.g., $\text{ReLU}(\cdot) = \max(0, \cdot)$), and $\mathbf{H}^l \in \mathbb{R}^{N \times F^l}$ is the matrix of activations in the l^{th} layer with $\mathbf{H}^0 = \mathbf{X}$, i.e., the input feature matrix of the data.

One indication of the effectiveness of the learned embeddings is that the distances between random pairs of anchor node embeddings are well distributed. To obtain the representation of such a network, we leverage a non-parameterized graph autoencoder for node embedding, inspired by [36]. Specifically, we use a two-layer neural networks to reconstruct the symmetrically normalized adjacency matrix as $\widetilde{\mathbf{W}} = \sigma(\mathbf{U}^\intercal \mathbf{U})$, where $\mathbf{U} \in \mathbb{R}^{N \times F}$ is the embedding learned by Eq.(4) and Eq.(5). After the training of reconstruction and obtaining optimized $\mathbf{U}$, we denote the social representation of each node/user by $\mathbf{u}_i$, i.e., the i^{th} row of $\mathbf{U}$. Then we define the cross-entropy loss function when training network embedding as follows:

$$\mathcal{L} = -\sum_{i=1}^{N}\sum_{j=1}^{N} \widehat{W}_{ij} \ln \widetilde{W}_{ij} \tag{5}$$

where $\widehat{W}_{ij} \in \widehat{\mathbf{W}}$ and $\widetilde{W}_{ij} \in \widetilde{\mathbf{W}}$.

Discriminative Regularization: The above GCN based graph representation model has the advantage of incorporating node labels for discriminating the nodes belonging to different classes. In our problem setting, the anchor nodes are more distinguishable in the latent space compared to the unsupervised network embedding methods, due to the wider distribution of the anchor nodes using the described semi-supervised discriminative embedding approach. However, there is a drawback: one cannot efficiently tell apart the anchor nodes from their neighborhoods. Here we present a regularization method combined with the graph kernel for further separating the embeddings of the nodes.

Intuitively, we need a high-resolution node representation and a suitable similarity measure operating on that representation. Formally, our node embeddings in the space $\mathbb{Z}$ are first projected into a high-dimensional feature space $\mathbb{F}$ via a function $\phi : \mathbb{Z} \to \mathbb{F}$, i.e., $\mathbf{U} \to \phi(\mathbf{U})$. The mapped embeddings in space $\mathbb{F}$ should not only be able to retain the distance information among previous representation of the nodes, but also to make the nodes in the space $\mathbb{F}$ more separated from each other.

There are many ways to measure the similarity between node embeddings in a high-dimensional feature space, but they typically involve calculating the inner product among nodes using their projected high-dimensional representation $\langle \phi(\mathbf{u}_i), \phi(\mathbf{u}_j) \rangle$, which would incur computational overheads. To address this issue, one can rely on the kernel trick to generalize distance-based algorithms to operate in the projected space [37] and the dot product can then be evaluated directly using a nonlinear function κ as $\kappa(\mathbf{u}_i, \mathbf{u}_j) = \langle \phi(\mathbf{u}_i), \phi(\mathbf{u}_j) \rangle$. Therefore, we define the distance between $\phi(\mathbf{u}_i)$ and $\phi(\mathbf{u}_j)$ in terms of the kernel as:

$$\begin{aligned} \mathcal{D}(\phi(\mathbf{u}_i), \phi(\mathbf{u}_j)) &= \|\phi(\mathbf{u}_i) - \phi(\mathbf{u}_j)\|^2 \\ &= \kappa(\mathbf{u}_i, \mathbf{u}_i) + \kappa(\mathbf{u}_j, \mathbf{u}_j) - 2\kappa(\mathbf{u}_i, \mathbf{u}_j) \end{aligned} \tag{6}$$

Although a kernel $\kappa(\mathbf{u}_i, \mathbf{u}_j)$ can be considered as a similarity measure between instances $\mathbf{u}_i$ and $\mathbf{u}_j$ in high-dimensional feature space $\mathbb{F}$, we are neither able to ensure that they are unit vectors nor to bound the value in the range of $(0, 1]$. Thus, we alternatively construct a similarity matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$ – each element $s_{ij} \in \mathbf{S}$ measures the similarity between $\phi(\mathbf{u}_i)$ and $\phi(\mathbf{u}_j)$ relying on the distance function specified in Eq.(6), and defined as:

$$s_{ij}(\phi(\mathbf{u}_i), \phi(\mathbf{u}_j)) = 1/\exp\{\mathcal{D}(\phi(\mathbf{u}_i), \phi(\mathbf{u}_j))\}, \tag{7}$$

which is now normalized to be in the range of $(0, 1]$ – the larger value of s_{ij}, the more similar between feature vectors $\mathbf{u}_i$ and $\mathbf{u}_j$.

Until now, anchor nodes have not been considered when embedding the network. However, it is interesting that our graph representation utilizes the role of anchor nodes – strictly speaking, the only label information that can be incorporated into node alignment are the anchor nodes. Thus, we use the representation of anchor nodes to reconstruct the representation of the remaining nodes. We first build an intermediate matrix $\mathbf{C} \in \mathbb{R}^{N \times F}$, whose i^{th} row $\mathbf{c}_i$ denotes the embeddings of anchor node u_i

$$\mathbf{c}_i = \begin{cases} \mathbf{u}_i, & \text{if } u_i \text{ is anchor node;} \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

This operation sets the non-anchor node representation as 0. Then, we use matrix $\mathbf{C}$ combined with above similarity matrix $\mathbf{S}$ to derive a new representation of nodes:

$$\mathbf{U}'_{N \times F} = \mathbf{S}_{N \times N} \cdot \mathbf{C}_{N \times F} \tag{9}$$

where i^{th} row of $\mathbf{U}'$ is a vector $\mathbf{u}'_i = \sum_{j=1}^{N} s_{ij} \cdot \mathbf{c}_j$.

We now explain the rationale behind this embedding reconstruction: (1) representing the non-anchor nodes using anchor ones can (partially) eliminate the bias of autoencoder based node embedding caused by non-anchor nodes – possible reason for confounding matching problem; (2) incorporating the similarity measure in Hilbert space (Eq.(7)) can help discriminate nodes that are non-separable in the original embedding space, which is also the reason of involving graph kernels; and (3) more importantly, reconstructing embedding can efficiently improve the matching performance, as we will demonstrate later.

B. Adversarial Matching

After obtaining the latent embedding space for two graphs $\mathcal{G}^s$ and $\mathcal{G}^t$, we turn to learn the mapping generator with known anchor nodes, which is a parameterized function implemented with neural networks in this work. Given a labeled anchor node pair (u_i, v_j) ($u_i \in \mathcal{V}^s$ and $v_j \in \mathcal{V}^t$) and their representation

vectors $(\mathbf{u}_i, \mathbf{v}_j)$, a mapping generator $\Phi(\mathbf{u}_i; \theta_1)$ is trained by minimizing the following loss function:

$$\ell(\mathbf{u}_i, \mathbf{v}_j) = \arg\min \left\{1 - \cos\left\langle \Phi\left(\mathbf{u}_i; \theta_1\right), \mathbf{v}_j \right\rangle\right\}, \tag{10}$$

Unfortunately, directly applying the mapping function may lead to large matching errors due to the confounding matching problem as illustrated in Sec. III. The main reason is that we only leverage the labeled anchor nodes for learning mapping function which incurs that all nodes (both anchor nodes and non-anchor ones) in $\mathcal{G}^s$ are towards approximating the distribution of known anchor nodes in $\mathcal{G}^t$ after mapping. Thus, unknown anchor nodes (e.g., those used in testing) are also departing from their true corresponding nodes in $\mathcal{G}^t$. Therefore, it is desirable to "distort" the non-training anchor nodes to be discriminable from the training ones while approaching their real corresponding nodes.

Towards that goal, we present an adversarial learning paradigm by leveraging the idea of Generative Adversarial Nets (GAN) [38], which have been primarily applied to generate real-like images, grammatically correct texts and fluent dialogue. Briefly, the generator in GAN produces an adversarial example to fool the discriminator, while the discriminator tries to distinguish against the adversary. However, our proposed matching method is no longer intended as a defense against an adversary, but as a means of regularizing the node matching by stabilizing the mapping function.

More formally, the adversarial matching consists of a *mapper* $\Phi(\cdot; \theta_1)$ and a *discriminator* $D(\cdot; \theta_2)$: $\Phi(\cdot; \theta_1)$ is a non-linear transformation of the vectors in $\mathbf{G}^s$ to vector representations in $\mathbf{G}^t$ which are regarded as fake samples, with the goal of trying to confuse the discriminator that a mapped vector $\mathbf{u}$ (sampled from a prior distribution $p_{\mathbf{u}}$, e.g., an uniform or a Gaussian) comes from the real anchor node distribution p_{anchor} in $\mathbf{G}^t$; simultaneously, $D(\cdot; \theta_2)$ computes the probability that a mapped vector $\mathbf{u}$ in target space is a sample from the real data distribution p_{anchor}, rather than from the data distribution p_m generated from our mapping function. This matching process is to learn the mapper's distribution p_m over anchor nodes $\mathbf{n}$ which can be considered as a minimax game with the mapper and discriminator playing against each other iteratively:

$$\begin{aligned}\min_{\Phi} \max_{G} \mathbb{E}_{\mathbf{n}\sim p_{\text{anchor}}(\mathbf{n})} \left[\log D\left(\mathbf{n}; \theta_2\right)\right] + \\ \mathbb{E}_{\mathbf{u}\sim p_{\mathbf{u}}(\mathbf{u})} \left[\log\left(1 - D\left(\Phi\left(\mathbf{u}; \theta_1\right); \theta_2\right)\right)\right]\end{aligned} \tag{11}$$

where $D(\cdot; \theta_2)$ adjusts its parameters θ_2 to maximize the capability of discriminating the mapped vectors from the real anchor nodes, while $\Phi(\cdot; \theta_1)$ minimizing $\log(1 - D(\Phi(\mathbf{u}; \theta_1); \theta_2))$ by tuning θ_1.

We note that adversarial matching is different from the discriminative embedding in terms of objective and methodology. Generally, adversarial matching is a method to learn the anchor node distribution in target space by reducing the empirical bias of training data. In contrast, we use discriminative embedding to widely distribute the anchor nodes and their neighborhoods in the latent space during embedding, although its outcome also benefits node matching in the later step. We argue and experimentally observe (Sec. VI) that above adversarial matching can successfully improve the network alignment performance especially for the *top-1* accuracy.

V. Matching Interpretability

Most of the neural network-based alignment methods are viewed as "black-box" models and limited by the lack of explaining the behaviors. To demystify the aligning behavior of *d*NAME, we estimate in a closed form the importance of each training sample (node) $\mathbf{u}$ on the NA performance of a particular testing instance $\mathbf{u}^{\text{test}}$ using the well-established theory from influence functions [18], [19].

Specifically, removing a data point $\mathbf{u}$ from the training set results in a change of $\theta^*_{-\mathbf{u}} - \theta^*$, where $\theta^*_{-\mathbf{u}}$ is the optimal parameter θ with the minimum total loss without the data point $\mathbf{u}$, denoted by: $\theta^*_{-\mathbf{u}} \stackrel{\text{def}}{=} \arg\min_{\theta\in\Theta} \sum_{\mathbf{u}_i \neq \mathbf{u}} \ell(\mathbf{u}_i, \theta)$, where $\ell(\mathbf{u_i}, , \theta)$ is the loss of aligning $\mathbf{u_i}$. To estimate the influence of each removed training sample $\mathbf{u}$ and avoid re-training the model, Koh et al. [29] use influence functions to efficiently approximate this behavior. The basic idea is to compute the parameter changes if $\mathbf{u}$ was upweighted by some small ϵ, which gives the new parameters $\theta^*_{\epsilon,\mathbf{u}} \stackrel{\text{def}}{=} \arg\min_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^{N} \ell(\mathbf{u}_i, \theta) + \epsilon\ell(\mathbf{u}, \theta)$. The influence of upweighting $\mathbf{u}$ on the parameters θ^* is given by

$$\mathcal{I}_{\text{up},\theta^*}(\mathbf{u}) \stackrel{\text{def}}{=} \left.\frac{\partial\theta^*_{\epsilon,\mathbf{u}}}{\partial\epsilon}\right|_{\epsilon=0} = -\mathbf{H}^{-1}_{\theta^*} \bigtriangledown_\theta \ell(\mathbf{u}_i, \theta^*) \tag{12}$$

where $\mathbf{H}_{\theta^*} = \frac{1}{N}\sum_{i=1}^{N} \bigtriangledown^2_\theta \ell(\mathbf{u}_i, \theta^*)$ is the Hessian matrix. Eq.(12) shows that removing $\mathbf{u}$ is the same as upweighting it by $\epsilon = -1/N$. Thus, one can linearly approximate the parameter change of removing $\mathbf{u}$ as $\theta^*_{-\mathbf{u}} - \theta^* \approx -\frac{1}{N}\mathcal{I}_{\text{up},\theta^*}(\mathbf{u})$ without re-training the model. The influence of upweighting a training point $\mathbf{u}$ on the loss for a testing point $\mathbf{u}^{\texttt{test}}$ can then be calculated according to the chain rule:

$$\begin{aligned}\mathcal{I}_{\text{up,loss}}(\mathbf{u}, \mathbf{u}^{\text{test}}) &\stackrel{\text{def}}{=} \left.\frac{\partial\ell(\mathbf{u}^{\text{test}}, \theta^*_{\epsilon,\mathbf{u}})}{\partial\epsilon}\right|_{\epsilon=0} \\ &= \bigtriangledown_\theta \ell(\mathbf{u}^{\text{test}}, \theta^*)^\intercal \left.\frac{\partial\theta^*_{\epsilon,\mathbf{u}}}{\partial\epsilon}\right|_{\epsilon=0} \\ &= -\bigtriangledown_\theta \ell(\mathbf{u}^{\text{test}}, \theta^*)^\intercal \mathbf{H}^{-1}_{\theta^*} \bigtriangledown_\theta \ell(\mathbf{u}, \theta^*).\end{aligned} \tag{13}$$

To speed up the computation, we use implicit Hessian-vector products (HVPs) to efficiently approximate $\mathcal{S}_{\text{test}} \stackrel{\text{def}}{=} \mathbf{H}^{-1}_{\theta^*} \bigtriangledown_\theta \ell(\mathbf{u}, \theta^*)$. Then, the $\mathcal{I}_{\text{up,loss}}(\mathbf{u}, \mathbf{u}^{\text{test}})$ can be rewritten as $\mathcal{I}_{\text{up,loss}}(\mathbf{u}, \mathbf{u}^{\text{test}}) = -\mathcal{S}_{\text{test}} \bigtriangledown_\theta \ell(\mathbf{u}, \theta^*)$. Since Hessian $\mathbf{H}_{\theta^*}$ is positive semidefinite by assumption, we have:

$$\mathbf{H}^{-1}_{\theta^*} \bigtriangledown_\theta \ell(\mathbf{u}, \theta^*) = \arg\min_{\nu} \{\frac{1}{2}\nu^\intercal \mathbf{H}_{\theta^*}\nu - \bigtriangledown_\theta \ell(\mathbf{u}, \theta^*)^\intercal \nu\}$$

where the exact solution ν can be obtained with conjugate gradients requiring only the evaluation of $\mathbf{H}_{\theta^*}\nu$ instead of explicitly computing $\mathbf{H}^{-1}_{\theta^*}$. We refer the reader to [29] for details and explanations on this topic, where the application of influence functions in computer vision is investigated. We will show the experimental results of explaining the network alignment and matching in Sec. VI.

VI. EXPERIMENTS

We now present in detail the experimental observations demonstrating the advantages of *d*NAME from two aspects: aligning precision and ranking performance. We also discuss the interpretability of our proposed model.

A. Datasets

To compare the performance of different methods, we use the following social networks collections in our experiments (cf. Table I):

- **Foursquare-Twitter** (F-T): This dataset is provided by Zhang et al. [34], where nodes (users) of two social networks (Foursquare and Twitter) are partially aligned.
- **Lastfm-MySpace** (L-M): This dataset is published by [12] and available online (http://aminer.org/cosnet). It contains 5 networks, however, for the sake of privacy, it only provides partial anchor nodes for true identity linkage.

TABLE I
STATISTICS OF DATASETS.

Dataset	$\|V\|$	$\|E\|$	# of anchor nodes
Foursquare	5,120	76,972	3,148
Twitter	5,313	164,920	
Lastfm	2,138	4,259	1,561
MySpace	2,117	3,798	

B. Baselines & Metrics

In this work, we use a 3-layer perceptron as the graph convolution to embed nodes into a 256-dimensional vectors with ReLU activation. As for the adversarial matching in *d*NAME, both the mapper and discriminator are 3-layer perceptrons. In addition, we also implement a simplified version of *d*NAME, called *d*NAME* with all components included except the adversarial matching. Both *d*NAME* and *d*NAME employ AMSGrad [39] as stochastic optimization for training the neural networks.

*d*NAME together with several network-based methods require only network structural information for alignment. We note that profile-related features can be incorporated to improve the performance. However, in this paper, we focus on network structure based NA and evaluate against the following baselines:

- **DeepLink** [6]: DeepLink is an end-to-end network alignment approach that samples networks and learns to encode nodes into vector representations to capture local and global network structures. A dual learning based paradigm is then employed to learn transferring knowledge and updating the anchor linkage with policy gradient.
- **IONE/ONE** [15]: Input-Output Network Embedding (IONE) is a network embedding and partial network alignment method. In IONE, the user latent space is obtained with negative sampling and constraints on common users of the networks, where gradient descent is used to train and align two networks with anchor nodes. ONE is a simplified version of IONE where only node vector and output vector representation of a user are considered for alignment.
- **MAH/MAG** [24]: Manifold Alignment on Hypergraph (MAH) is the network embedding method which represents nodes into a common low dimensional space and infers account correlation by comparing distances between two vectors across networks in the embedding space. Manifold Alignment on traditional Graphs (MAG) builds a social graph for each network by computing user-to-user pairwise weights.
- **CRW**: Collective Random Walk (CRW) [27] predicts the formation of social links among users in the target network as well as anchor links aligning the target network with other external social networks.

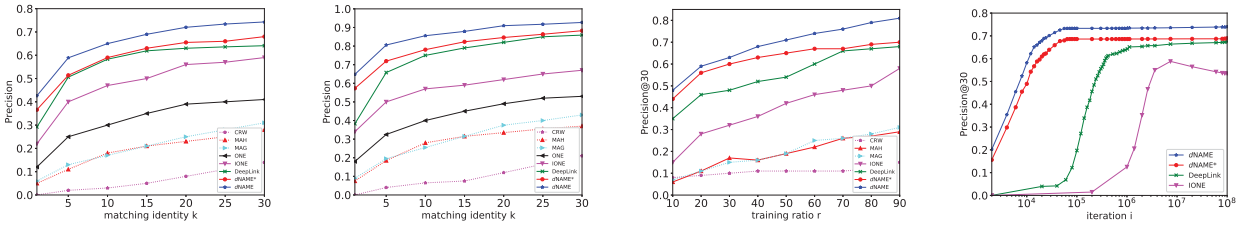
To evaluate the linking accuracy (inline with some of the existing works [2], [6]), we use a rather standard evaluation metric – *Precision@k(P@k)*, which is the portion of the relevant items from among the top k recommended ones (note that the higher the value, the better the performance). We omit the results on ranking performance comparison due to the space limitation. However, we visualize different methods on alignment results which, in a sense, reflects their ranking performance.

$$P@k = \sum_{i}^{n} \mathbb{1}_i\{success@k\}/n \tag{14}$$

where $\mathbb{1}_i\{success@k\}$ incidates whether the positive matching identity exists in the *top-k* ($k <= n$) list, and n is the number of testing anchor nodes. Note that since *top-k* is a metric of the true positive prediction rate, *Precision@k* is exactly the same as *Recall@k*, as well as the $F_1@k$, in the context of network alignment.

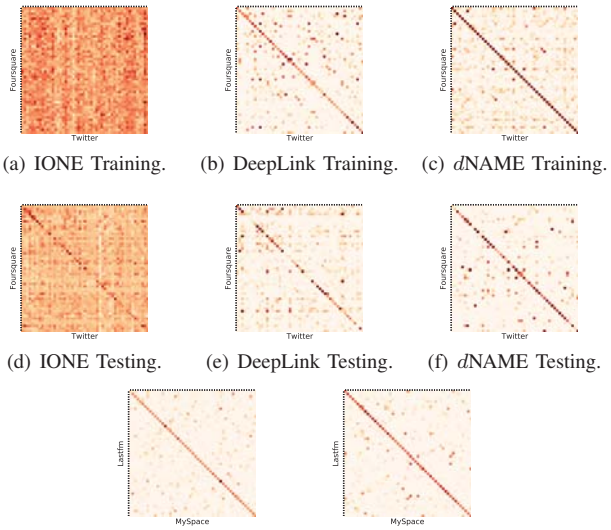
C. Results

Network Alignment: We first systematically evaluate various methods on the *top-k* accuracy of NA. As shown in Fig. 4(a) and 4(b) for the precision over different values of parameter k on two datasets, *d*NAME consistently outperforms the baselines. Compared to DeepLink – the best approach in baselines, it achieves 17% and 53% higher (on average) over the best DeepLink in terms of *top-30* and *top-1* accuracy, respectively. This is an empirical evidence that disentangled network embedding and de-confounding matching are effective for the NA problem. We also observe that *d*NAME outperforms *d*NAME* by 10% on *top-30* and 16% on *top-1* accuracy, respectively, demonstrating that our adversarial matching can further improve the alignment performance by tackling the confounding matching with a minimax game. The remaining methods (CRW, MAG, and IONE), yield significantly lower scores than our approach. This is due to the fact that the crux of these methods is to learn heterogeneous embedding without addressing the contradictory objectives between embedding and matching. We also note that *d*NAME significantly improves the *top-1* alignment performance – critical for applications seeking higher Recall scores. While Figure 4 only shows results on Precision vs. k for Lastfm-MySpace, the rest of comparisons on this dataset – omitted for space limitation - are consistent.



(a) Precision vs. k (F-T). (b) Precision vs. k (L-M). (c) Precision vs. r (F-T). (d) Precision vs. i (F-T).

Fig. 4. Alignment precision results. Parameters: k is the predicted k top matching identities; r is the percentage of anchor nodes used for training; i is i^{th} training iteration.

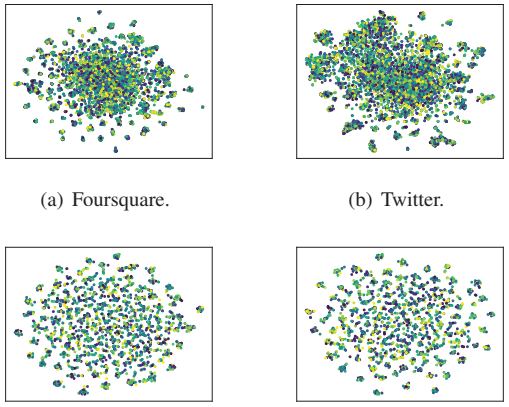


(a) IONE Training. (b) DeepLink Training. (c) *d*NAME Training.

(d) IONE Testing. (e) DeepLink Testing. (f) *d*NAME Testing.

(g) *d*NAME Training. (h) *d*NAME Testing.

Fig. 5. Heat map/alignment comparison: Plot the matching scores of training the top 50 accurate aligned nodes for each algorithm. (a) - (c): training on Foursquare-Twitter. (d) - (f): testing on Foursquare-Twitter dataset. (g) - (h): training and test of *d*Name on Lastfm-MySpace dataset.



(a) Foursquare. (b) Twitter.

(c) Lastfm. (d) MySpace.

Fig. 6. Latent space visualization of *d*NAME on different datasets.

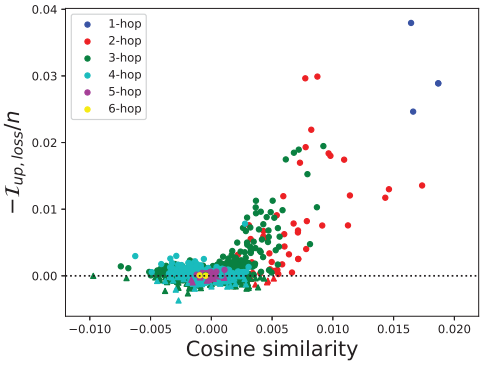


Fig. 7. Impact of node matching from training samples: Nodes above (below) the dashed horizontal line have positive (negative) impact on the alignment.

There is a trade-off when training a supervised model: – with a larger training sample, the model is better at capturing informative patterns, but it incurs greater computational cost. Fig. 4(c) plots the performance of the algorithms as a function of the size of training sample (represented by the training ratio r). It shows that *d*NAME, as well as *d*NAME*, requires less training data (anchor nodes) for boosting the models. This is an advantage of the graph convolution employed in our approach, i.e., it requires less labeled data to learn the network structure and can inductively propagate the node information by local convolutional operation.

Efficiency (convergence rate) is another important metric to consider when training networks. As shown in Fig. 4(d), both *d*NAME and *d*NAME* converge very fast compared to DeepLink and IONE – we do not consider the remaining methods due to their inferior performance (note the exponential scale of iterations in the figure). The result also demonstrates that our discriminative embedding with kernel trick is more efficient than the methods directly calculating the vector similarity in projected high-dimensional space, and that adversarial matching explored by *d*NAME successfully stabilizes the node alignment process. In addition, we find that IONE suffers from overfitting while *d*NAME and DeepLink are more robust to the increase of the training iterations.

Visualization. We use heat maps to plot the alignment performance (for both training and testing sets) for IONE, DeepLink and *d*NAME, as shown in Fig. 5. The darkness of a block corresponds to the similarity score of matching nodes and the darkness is consistent for all comparisons. Clearly, our

approach achieves more discriminative results (darker along the diagonals) for both training and testing. We note that this visualization indirectly reflects the ranking performance. Further, we observe that *d*NAME achieves more widely distributed embedding compared to DeepWalk and node2vec (cf. Fig. 2), due to its discriminative graph convolution, as shown in Fig. 6. This higher-resolution representation manifests that we bridge the gap between the graph kernel based technique and the network alignment applications.

Matching Explanation. Finally, we explain the aligning behavior by first randomly selecting a testing anchor node pair that has been successfully aligned. Then we measure how much impact (both positive and negative) of all training samples on successfully aligning the selected anchor node pair, as shown in Fig. 7. Additionally, the closer training sample (e.g., 1-hop) to the testing anchor node before mapping, the more positive impact it has. Although such a fact that neighborhoods plays more important role on representation and aligning a particular anchor is intuitive, we are the first proving it and quantifying its influence.

VII. Conclusions and Future Works

We presented the *d*NAME – a novel approach for disentangled networks embedding. Our discriminative matching algorithm addressed the network alignment problem by leveraging the graph convolution for semi-supervised embedding and graph kernels to distinguish anchor nodes from their neighborhoods. *d*NAME is an adversarial node aligning approach capable of addressing the confounding matching problem inherent in existing methods. In addition to the new methodology that ensures higher network alignment performance, we introduced the use of influence functions which, when it comes to training, successfully explain the network alignment behavior and provide insight of the impact of all training samples. This, in turn, can be beneficial not only for network alignment but also other identity-based applications.

VIII. Acknowledgements

Work supported by National Natural Science Foundation of China (Grants No.61602097 and No.61472064), NSF Grants III 1213038 and CNS 1646107, ONR grant N00014-14-10215.

References

[1] C.-S. Liao, K. Lu, M. Baym, R. Singh, and B. Berger, "Isorankn: spectral methods for global alignment of multiple protein networks," *Bioinformatics*, 2009.

[2] K. Shu, S. Wang, J. Tang, R. Zafarani, and H. Liu, "User identity linkage across online social networks: A review," *SIGKDD Explorations Newsletter*, vol. 18, no. 2, pp. 5–17, 2017.

[3] M. Jiang, P. Cui, N. J. Yuan, and X. Xie, "Little is much: Bridging cross-platform behaviors through overlapped crowds," in *AAAI*, 2016.

[4] O. Goga, H. Lei, S. H. K. Parthasarathi, G. Friedland, R. Sommer, and R. Teixeira, "Exploiting innocuous activity for correlating users across sites," in *WWW*, 2013.

[5] X. Mu, F. Zhu, E. P. Lim, J. Xiao, J. Wang, and Z. H. Zhou, "User identity linkage by latent user space modelling," in *SIGKDD*, 2016.

[6] F. Zhou, L. Liu, K. Zhang, G. Trajcevski, J. Wu, and T. Zhong, "Deeplink: A deep learning approach for user identity linkage," in *INFOCOM*, 2018.

[7] N. Korula and S. Lattanzi, "An efficient reconciliation algorithm for social networks," in *VLDB*, 2014.

[8] S. Liu, S. Wang, F. Zhu, J. Zhang, and R. Krishnan, "Hydra: large-scale social identity linkage via heterogeneous behavior modeling," in *SIGMOD*, 2014.

[9] X. Kong, J. Zhang, and P. S. Yu, "Inferring anchor links across multiple heterogeneous social networks," in *CIKM*, 2013.

[10] Y. Shen and H. Jin, "Controllable information sharing for user accounts linkage across multiple online social networks," in *CIKM*, 2014.

[11] J. Zhang, J. Chen, S. Zhi, Y. Chang, P. S. Yu, and J. Han, "Link prediction across aligned networks with sparse and low rank matrix estimation," in *ICDE*, 2017.

[12] Y. Zhang, J. Tang, Z. Yang, J. Pei, and P. S. Yu, "Cosnet: Connecting heterogeneous social networks with local and global consistency," in *SIGKDD*, 2015.

[13] S. Zhang and H. Tong, "Final: Fast attributed network alignment," in *SIGKDD*, 2016.

[14] H. Cai, V. W. Zheng, and K. Chang, "A comprehensive survey of graph embedding: problems, techniques and applications," *TKDE*, 2018.

[15] L. Liu, W. K. Cheung, X. Li, and L. Liao, "Aligning users across social networks using network embedding," in *IJCAI*, 2016.

[16] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*, 2014.

[17] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*, 2016.

[18] R. D. Cook and S. Weisberg, "Characterizations of an empirical influence function for detecting influential cases in regression," *Technometrics*, 1980.

[19] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel, *Robust Statistics: The Approach Based on Influence Functions*. Wiley, 2005.

[20] T. Man, H. Shen, S. Liu, X. Jin, and X. Cheng, "Predict anchor links across social networks via an embedding approach," in *IJCAI*, 2016.

[21] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *JMLR*, 2010.

[22] S. Ghosh, N. Das, T. Gonçalves, P. Quaresma, and M. Kundu, "The journey of graph kernels through two decades," *Computer Science Review*, vol. 27, pp. 88–111, 2018.

[23] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *JMLR*, 2011.

[24] S. Tan, Z. Guan, D. Cai, X. Qin, J. Bu, and C. Chen, "Mapping users across networks by manifold alignment on hypergraph," in *AAAI*, 2014.

[25] R. Zafarani, L. Tang, and H. Liu, "User identification across social media," *ACM Trans. Knowl. Discov. Data*, vol. 10, no. 2, 2015.

[26] X. Zhou, X. Liang, H. Zhang, and Y. Ma, "Cross-platform identification of anonymous identical users in multiple social media networks," *IEEE TKDE*, vol. 28, no. 2, pp. 411–424, 2016.

[27] J. Zhang and P. S. Yu, "Integrated anchor and social link predictions across social networks," in *IJCAI*, 2015.

[28] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.

[29] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *ICML*, 2017.

[30] C. Villani, "Topics in optimal transportation," *Graduate studies in mathematics*, 2003.

[31] F. D. Johansson and D. P. Dubhashi, "Learning with similarity functions on graphs using matchings of geometric embeddings," in *SIGKDD*, 2015.

[32] G. Nikolentzos, P. Meladianos, and M. Vazirgiannis, "Matching node embeddings for graph similarity," in *AAAI*, 2017.

[33] H. Zou, "Distance, dissimilarity index, and network community structure," *PHYSICAL REVIEW E*, vol. 67, 2003.

[34] J. Zhang and P. S. Yu, "Pct: Partial co-alignment of social networks," in *WWW*, 2016.

[35] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[36] ——, "Variational graph auto-encoders," *arxiv*, 2016.

[37] B. Schölkopf, "The kernel trick for distances," in *NIPS*, 2000.

[38] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014.

[39] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *ICLR*, 2018.