Balancing Consistency and Disparity in Network Alignment

Si Zhang*, Hanghang Tong*, Long Jin†, Yinglong Xia†, Yunsong Guo†
* University of Illinois at Urbana-Champaign, {sizhang2, htong}@illinois.edu
† Facebook, {longjin, yxia, yunsong}@fb.com

ABSTRACT

Network alignment plays an important role in a variety of applications. Many traditional methods explicitly or implicitly assume the *alignment consistency* which might suffer from over-smoothness, whereas some recent embedding based methods could somewhat embrace the *alignment disparity* by sampling negative alignment pairs. However, under different or even competing designs of negative sampling distributions, some methods advocate positive correlation which could result in false negative samples incorrectly violating the alignment consistency, whereas others champion negative correlation or uniform distribution to sample nodes which may contribute little to learning meaningful embeddings. In this paper, we demystify the intrinsic relationships behind various network alignment methods and between these competing design principles of sampling. Specifically, in terms of model design, we theoretically reveal the close connections between a special graph convolutional network model and the traditional consistency based alignment method. For model training, we quantify the risk of embedding learning for network alignment with respect to the sampling distributions. Based on these, we propose NeXtAlign which strikes a balance between alignment consistency and disparity. We conduct extensive experiments that demonstrate the proposed method achieves significant improvements over the state-of-the-arts.

CCS CONCEPTS

• **Information systems** → **Data mining**; • **Computing methodologies** → **Machine learning**.

KEYWORDS

Network alignment; negative sampling; network embedding

ACM Reference Format:

Si Zhang, Hanghang Tong, Long Jin, Yinglong Xia, Yunsong Guo. 2021. Balancing Consistency and Disparity in Network Alignment. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467331

1 INTRODUCTION

In the age of big data, multiple networks emerge in many influential domains, such as social networks of different online social platforms and protein-protein interaction (PPI) networks of different species. Network alignment, which aims to uncover the node correspondences across networks (e.g., dashed lines in Figure 1 (a)), plays a crucial role in distilling values from multiple networks. Specifically, with the node correspondences, multiple networks can be integrated into a world-view network which may exhibit the patterns that are invisible if mining networks separately. For example, aligning proteins across PPI networks of different species facilitates to transfer knowledge from well-studied species to less-studied species and explore the evolutionary relationships [7]. In addition, by integrating multiple transaction networks, the complex fraud behaviors (e.g., money laundering) that are covert in each individual network can be brought to light.

Despite extensive research on network alignment, many traditional methods explicitly or implicitly assume the *alignment consistency* [21, 31, 32]. That is, the alignments of neighboring node pairs tend to be consistent with each other. For example, FINAL [32] explicitly formulates the alignment consistency as minimizing the differences between the alignment of two nodes and those of their corresponding neighbors. Alternatively, the objective can be interpreted as directly generating positive alignment pairs by *all* neighboring node pairs (e.g., (c, f) from the anchor link (b, y)) and then minimizing the differences among them. However, optimizing this alignment consistency might result in the over-smoothness issue of the alignments within a local neighborhood and fail to distinguish the correct alignments from misleading alignments.

On the other hand, embedding based methods, which infer node alignments by node embeddings, can to some extent incorporate the *alignment disparity* and ameliorate the over-smoothness issue by bringing in the negative alignment pairs. For example, some alignment methods (such as [3, 13]) sample negative alignment pairs by a degree-based sampling distribution and learn node embeddings that classify the sampled alignment pairs into the negative class. Others apply a uniform distribution to randomly sample negative alignment pairs used in the ranking loss [22, 26]. With negative sampling, the learned node embeddings can potentially make the alignments within the local neighborhood more separable (i.e., alignment disparity). More recently, negative sampling for *single* network embedding has been riveted and various sampling strategies are proposed under different or even competing design principles. For example, [28] advocates a positive correlation between negative and positive sampling distributions. That is, nodes with similar embeddings are more likely to be sampled such that *hard* negative samples are encouraged. Others exclusively favor a negative correlation to better preserve local proximity [1, 14].

Despite various negative sampling strategies, it still remains opaque how negative sampling should be designed to benefit network alignment. In other words, a more fundamental question is

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467331

what makes a 'good' negative pair in the context of network alignment? Intuitively, good negative samples should distinguish the anchor links and the close node pairs that may mislead the alignment, while not violating the overall alignment consistency. In addition, these negative samples should inform the model to learn meaningful embeddings for network alignment. However, the aforementioned negative sampling strategies have their own limitations. To be specific, owing to its root in single network embedding, positive correlation based sampling [28] may lead to the false negative alignment pairs (e.g., (c, f) as the negative alignment pair of anchor link (b, y) in Figure 1 (a)) which results in incorrect alignments violating the alignment consistency. On the other side, using the pre-defined distributions [13, 22] and those that are negatively correlated to the positive sampling distribution [1, 14] may sample distant or dissimilar node pairs (e.g., (e, h)) that may not contribute much to learning meaningful node embeddings.

In this paper, we strive to demystify the intrinsic relationships behind different competing designs for network alignment (i.e., alignment consistency vs. disparity, negative correlation vs. positive or neutral correlation), so that we can strike a good trade-off between them. We address this from both the model architecture and model training perspectives. First (*model design*), we theoretically prove that the alignments inferred by graph convolutional networks resemble the semi-supervised variant of the consistency based alignment method FINAL [32]. This inspires a specific graph convolutional network model that can preserve alignment consistency. Second (*model training*), we provide a lemma that implies the mean square error between the inner products of node embeddings learned by expected loss and empirical loss can be quantified by different sampling distributions. To reduce the error while making the aforementioned competing designs compatible with each other, we design a novel alignment scoring function which paves the way to the proposed sampling strategies. Armed with these components, we propose a novel semi-supervised method that accommodates both alignment consistency and alignment disparity. The main contributions of the paper are summarized as below.

- **Problem Definition.** To our best knowledge, we are the first to study the trade-off between the consistency and disparity in network alignment.
- **Method and Analysis.** We theoretically reveal the close connections between graph convolutional networks and consistency based alignment method and the intrinsic relationships behind the competing principles of sampling designs. Based on them, we propose a new semi-supervised network alignment method NeXtAlign.
- **Empirical Evaluations.** Extensive experiments validate significant improvements compared to the state-of-the-arts.

The rest of the paper is organized as follows. Section 2 defines the semi-supervised network alignment problem and introduces the preliminaries. Section 3 presents the proposed network alignment method. Section 4 shows the experimental results. Related works and conclusion are presented in Section 5 and Section 6.

2 PROBLEM DEFINITION

Table 1 summarizes the main notations used in the paper. We use bold uppercase letters for matrices (e.g., $\mathbf{L}$), bold lowercase letters for vectors (e.g., $\mathbf{a}$), lowercase letters (e.g., α) for scalars and uppercase calligraphic letters (e.g., $\mathcal{L}$) for sets. We denote the transpose by a superscript prime (e.g., $\mathbf{L}'$ as the transpose of $\mathbf{L}$).

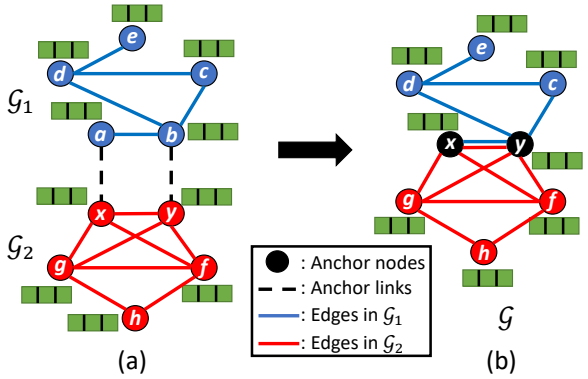


Figure 1: The world-view network.

2.1 Semi-Supervised Network Alignment

We denote the input networks by $\mathcal{G}_1 = \{\mathcal{V}_1, \mathbf{A}_1, \mathbf{X}_1\}$ and $\mathcal{G}_2 = \{\mathcal{V}_2, \mathbf{A}_2, \mathbf{X}_2\}$ where $\mathcal{V}_1, \mathcal{V}_2$ represent the node sets, $\mathbf{A}_1, \mathbf{A}_2$ represent the adjacency matrices and $\mathbf{X}_1, \mathbf{X}_2$ represent the input node attributes of $\mathcal{G}_1, \mathcal{G}_2$ respectively. And we denote $n_1 = |\mathcal{V}_1|, n_2 = |\mathcal{V}_2|$ as the number of nodes in two networks and the input attribute matrices are of sizes $\mathbf{X}_1 \in \mathbb{R}^{d_0 \times n_1}, \mathbf{X}_2 \in \mathbb{R}^{d_0 \times n_2}$. In addition, anchor links are defined as the node pairs that are one-to-one mapped a priori. For example, given a set of anchor links $\mathcal{L} = \{(a, x) | a \in \mathcal{V}_1, x \in \mathcal{V}_2\}$, it indicates that node-a in $\mathcal{G}_1$ is aligned with node-x in $\mathcal{G}_2$ a priori. In this way, node-a is called as an anchor node in $\mathcal{G}_1$ and node-x is an anchor node in $\mathcal{G}_2$. We also define $\mathcal{L}_1 = \{a | \exists x \in \mathcal{V}_2, s.t., (a, x) \in \mathcal{L}\}$ as the set of anchor nodes in $\mathcal{G}_1$ and similarly $\mathcal{L}_2$ for $\mathcal{G}_2$. Accordingly, the sets of non-anchor nodes are denoted by $\bar{\mathcal{L}}_1 = \mathcal{V}_1 - \mathcal{L}_1$ and $\bar{\mathcal{L}}_2 = \mathcal{V}_2 - \mathcal{L}_2$. As for indexing nodes, we use b, y as the general indices to index all the nodes in $\mathcal{V}_1$ and $\mathcal{V}_2$. In addition, we use a, x to specifically index the anchor nodes in $\mathcal{L}_1, \mathcal{L}_2$, and use u, v to index non-anchor nodes.

Given above notation definitions, our goal is to learn node embeddings and infer the alignment matrix $\mathbf{S} \in \mathbb{R}^{n_1 \times n_2}$. Formally, we define the semi-supervised network alignment as follows.

Table 1: Symbols and Notations

Symbols	Definition
$\mathcal{G}_1, \mathcal{G}_2$	input networks
$\mathcal{L}$	the set of anchor links across $\mathcal{G}_1, \mathcal{G}_2$
$\mathcal{L}_1, \mathcal{L}_2$	the sets of corresponding anchor nodes in $\mathcal{G}_1$ and $\mathcal{G}_2$
$\bar{\mathcal{L}}_1, \bar{\mathcal{L}}_2$	the sets of non-anchor nodes in $\mathcal{G}_1$ and $\mathcal{G}_2$
a, x	indices of anchor nodes in $\mathcal{G}_1$ and $\mathcal{G}_2$
u, v	indices of non-anchor nodes in $\mathcal{G}_1$ and $\mathcal{G}_2$
b, y	indices of all nodes in $\mathcal{G}_1$ and $\mathcal{G}_2$
$\mathbf{a}, \mathbf{x}$	column embedding vectors of node-a and node-x
$[\cdot \| \cdot]$	vertical concatenation of two column vectors
$\odot$	Hadamard product

Problem 1. *Semi-Supervised Network Alignment.*

Given: *(1) undirected networks* $\mathcal{G}_1 = \{\mathcal{V}_1, \mathbf{A}_1, \mathbf{X}_1\}$ *and* $\mathcal{G}_2 = \{\mathcal{V}_2, \mathbf{A}_2, \mathbf{X}_2\}$*, and (2) a set of anchor links* $\mathcal{L}$*.*

Output: *the alignment matrix* $\mathbf{S}$ *where* $\mathbf{S}(a, x)$ *indicates how likely node-a and node-x are aligned.*

Note that for networks without node attributes, we use the one-hot encoding of a node as its input node attributes [20]. Given the fact that the nodes of an anchor link essentially represent the same entity, we can integrate the input networks $\mathcal{G}_1, \mathcal{G}_2$ into a world-view network $\mathcal{G}$ by merging two anchor nodes into a single node. As a result, the world-view network $\mathcal{G}$ has (1) non-anchor nodes in $\mathcal{G}_1, \mathcal{G}_2$ and unique anchor nodes as the nodes of $\mathcal{G}$, and (2) all edges of $\mathcal{G}_1, \mathcal{G}_2$ co-exist in $\mathcal{G}$ (shown in Figure 1 (b)). By learning node embeddings in this world-view network $\mathcal{G}$, we can naturally share the unique embedding across two corresponding anchor nodes, i.e., $\mathbf{a} = \mathbf{x},\ \forall (a, x) \in \mathcal{L}$. In the paper, we use $\mathbf{a}$ and $\mathbf{x}$ interchangeably.

2.2 Preliminaries

In many existing network alignment methods, a widely used assumption is the *alignment consistency* assumption that neighboring node pairs tend to have consistent alignments [2, 21, 32]. The existing method FINAL [32] formulates this assumption as

$$\min_{\mathbf{S}} \sum_{a,b,x,y} \left[\frac{\mathbf{S}(a,x)}{\sqrt{|\mathcal{N}_1(a)||\mathcal{N}_2(x)|}} - \frac{\mathbf{S}(b,y)}{\sqrt{|\mathcal{N}_1(b)||\mathcal{N}_2(y)|}} \right]^2 \mathbf{A}_1(a,b)\mathbf{A}_2(x,y) \tag{1}$$

where $\mathcal{N}_1(a), \mathcal{N}_2(x)$ denote the neighbors of node-a in $\mathcal{G}_1$ and node-x in $\mathcal{G}_2$. Besides, we have $\mathbf{A}_1 = \mathbf{A}_1', \mathbf{A}_2 = \mathbf{A}_2'$. Here, Eq. (1) encourages a small difference between $\mathbf{S}(a,x)$ and $\mathbf{S}(b,y)$ if node-b and node-y are close neighbors of node-a and node-x. In the meanwhile, we can interpret Eq. (1) from another angle. That is, given an anchor link (a,x) with a large $\mathbf{S}(a,x)$, it encourages $\mathbf{S}(b,y)$ to be consistent with $\mathbf{S}(a,x)$ (i.e., large $\mathbf{S}(b,y)$ as well), which means the objective function in Eq. (1) naturally considers all the neighboring node pairs as the positive alignment pairs of (a,x). To solve Eq. (1), a fixed-point update in the t-th iteration is computed as

$$\mathbf{S}^t = \tilde{\mathbf{A}}_1 \mathbf{S}^{t-1} \tilde{\mathbf{A}}_2' \tag{2}$$

where $\tilde{\mathbf{A}}_1, \tilde{\mathbf{A}}_2$ are the symmetric normalization of $\mathbf{A}_1, \mathbf{A}_2$.

In the semi-supervised setting, the supervision (i.e., anchor links) can be used as a regularization upon Eq. (2), which leads to

$$\mathbf{S}^t = \alpha \tilde{\mathbf{A}}_1 \mathbf{S}^{t-1} \tilde{\mathbf{A}}_2' + (1-\alpha)\mathbf{L} \tag{3}$$

where α controls the importance of the alignment consistency. $\mathbf{L}(a,:) = 0$ and $\mathbf{L}(:,x) = 0$ except $\mathbf{L}(a,x) = 1$ if and only if $(a,x) \in \mathcal{L}$.

3 METHOD

3.1 Model Overview and Key Ideas

The core challenge that we aim to address is to design and train the model to strike a balance between the alignment consistency and disparity. To design the model that learns node embeddings while encouraging alignment consistency, we first prove that the alignments inferred by the node embeddings by a specific message passing without parameters (Eq. (5)) resemble the semi-supervised FINAL [32]. The key idea of the proof is to conduct a rank-$|\mathcal{L}|$ decomposition on matrix $\mathbf{L}$ used in Eq. (3) and use the decomposed matrices as the input node embeddings. Intuitively, by viewing the anchor nodes as the landmarks in the $|\mathcal{L}|$-dimensional Euclidean space, this message passing can be interpreted as to determine the relative positions for all nodes w.r.t. the anchor nodes. Next, based on its capability of capturing alignment consistency, we propose the parameterized counterpart of this message passing in Eq. (6). We name it as the RelGCN layer which is then used to calibrate the relative positions of nodes calculated by Eq. (5). The final node embeddings are obtained by applying a linear layer on these position vectors. The overall architecture is shown in Figure 2.

In terms of model training, the key idea of achieving the trade-off is by different sampling distributions. The intuitions behind it are as follows. Given an anchor link $(a,x) \in \mathcal{L}$, if (b,y) is sampled as the positive alignment pair, it encourages the consistency between node pairs (b,y) and (a,x). In contrast, if (b,y) is sampled as the negative alignment pair, the alignment disparity is favored between them. Besides, to preserve the local proximity within the same network, we also sample positive context pairs and negative context pairs. To design these sampling distributions, we first quantify the mean square error between the inner products of node embeddings learned by minimizing the expected loss and empirical loss. Based on this, to make the inner products of the high-probability node pairs to be estimated more accurately while satisfying different and even competing design principles, we compose a novel alignment scoring function that reflects multiple aspects of node embeddings.

3.2 Model Design

We first connect the fixed-point update of FINAL (i.e., Eq. (2)) to the vanilla GCN [11]. Suppose the alignment of nodes a, x is computed by $\mathbf{S}(a,x) = \mathbf{a}'\mathbf{x}$, then we have the following update

$$\begin{aligned}(\mathbf{a}^t)'\mathbf{x}^t = \mathbf{S}^t(a,x) &= \tilde{\mathbf{A}}_1(a,:)\mathbf{S}^{t-1}\tilde{\mathbf{A}}_2(:,x) \\ &= \sum_{b\in\mathcal{N}_1(a)} \sum_{y\in\mathcal{N}_2(x)} \frac{(\mathbf{b}^{t-1})'\mathbf{y}^{t-1}}{\sqrt{|\mathcal{N}_1(a)||\mathcal{N}_1(b)||\mathcal{N}_2(x)||\mathcal{N}_2(y)|}} \\ &= \sum_{b\in\mathcal{N}_1(a)} \frac{(\mathbf{b}^{t-1})'}{\sqrt{|\mathcal{N}_1(a)||\mathcal{N}_1(b)|}} \sum_{y\in\mathcal{N}_2(x)} \frac{\mathbf{y}^{t-1}}{\sqrt{|\mathcal{N}_2(x)||\mathcal{N}_2(y)|}}\end{aligned} \tag{4}$$

where $\mathbf{b}^{t-1}$ represents the node embedding of node-b in the $(t-1)$-th iteration/layer. As we can see, the t-th iteration of computing the alignment $\mathbf{S}^t(a,x)$ is equivalent to updating the node embeddings by applying the vanilla GCN without parameters

$$\mathbf{a}^t = \sum_{b\in\mathcal{N}_1(a)} \frac{\mathbf{b}^{t-1}}{\sqrt{|\mathcal{N}_1(a)||\mathcal{N}_1(b)|}},\ \mathbf{x}^t = \sum_{y\in\mathcal{N}_2(x)} \frac{\mathbf{y}^{t-1}}{\sqrt{|\mathcal{N}_2(x)||\mathcal{N}_2(y)|}}$$

followed by the inner product. Note that different from the vanilla GCN, we consider the neighborhood without self-loops. In addition, due to the over-smoothness issue of GCNs [12], the node alignments inferred by the node embeddings above could also suffer from over-smoothness, which might hamper the performance.

In the semi-supervised setting where anchor links are available, we design the following message passing without parameters

$$\begin{aligned}\mathbf{u}^t &= \sqrt{\alpha} \sum_{b\in\mathcal{N}_1(u)} \frac{\mathbf{b}^{t-1}}{\sqrt{|\mathcal{N}_1(u)||\mathcal{N}_1(b)|}} + \sqrt{1-\alpha}\mathbf{u}^{t-1} \\ \mathbf{v}^t &= \sqrt{\alpha} \sum_{y\in\mathcal{N}_2(v)} \frac{\mathbf{y}^{t-1}}{\sqrt{|\mathcal{N}_2(v)||\mathcal{N}_2(y)|}} + \sqrt{1-\alpha}\mathbf{v}^{t-1} \\ \mathbf{a}^t = \mathbf{x}^t &= \sqrt{\alpha} \sum_{b\in\mathcal{N}_1(a)} \frac{\mathbf{b}^{t-1}}{\sqrt{|\mathcal{N}_1(a)||\mathcal{N}_1(b)|}} + \sqrt{1-\alpha}\mathbf{x}^{t-1} \\ &\quad + \sqrt{\alpha} \sum_{y\in\mathcal{N}_2(x))} \frac{\mathbf{y}^{t-1}}{\sqrt{|\mathcal{N}_2(x)||\mathcal{N}_2(y)|}}\end{aligned} \tag{5}$$

where node-u, node-v are non-anchor nodes and a, x are anchor nodes. As shown in Lemma 1, the alignments inferred by embeddings in Eq. (5) resemble the semi-supervised FINAL (i.e., Eq. (3)).

Lemma 1. *Suppose the initial non-anchor node embeddings are* $\mathbf{u}^0 = \mathbf{v}^0 = \mathbf{0}$ *and those of the anchor nodes are* $\mathbf{a}^0 = \mathbf{x}^0 = \mathbf{e}_i \in \mathbb{R}^{|\mathcal{L}|}$ *where* (a, x) *is the* i*-th anchor link,* $\mathbf{e}_i(i) = 1$ *and* $\mathbf{e}_i(j) = 0,\ \forall j \neq i$*. Then by updating Eq.* (5) *once, the alignments computations are equivalent to Eq.* (3) *up to additional intra-network proximity and possible scaling terms.*

Proof. Given $|\mathcal{L}| = L$ anchor links, we can conduct a rank-L decomposition upon $\mathbf{L}$ without errors into $\mathbf{L} = \mathbf{L}_1'\mathbf{L}_2$. Since $\mathbf{L}(a, x) = 1$ if $(a, x) \in \mathcal{L}$, we have $\mathbf{L}_1(:, a) = \mathbf{a}^0 = \mathbf{e}_i$ and $\mathbf{L}_2(:, x) = \mathbf{x}^0 = \mathbf{e}_i$. For non-anchor nodes, we have $\mathbf{L}_1(:, u) = \mathbf{u}^0 = \mathbf{0}$, $\mathbf{L}_2(:, v) = \mathbf{v}^0 = \mathbf{0}$. After initializing embeddings as $\mathbf{L}_1, \mathbf{L}_2$, the alignments are computed by the inner products among the updated embeddings with Eq. (5).

$$\mathbf{S}(u, v) = \alpha \sum_{b \in \mathcal{N}_1(u)} \frac{(\mathbf{b}^0)'}{\sqrt{|\mathcal{N}_1(u)||\mathcal{N}_1(b)|}} \sum_{y \in \mathcal{N}_2(v)} \frac{\mathbf{y}^0}{\sqrt{|\mathcal{N}_2(v)||\mathcal{N}_2(y)|}}$$

$$\begin{aligned}\mathbf{S}(u, x) = &\alpha \sum_{b \in \mathcal{N}_1(u)} \frac{(\mathbf{b}^0)'}{\sqrt{|\mathcal{N}_1(u)||\mathcal{N}_1(b)|}} \sum_{y \in \mathcal{N}_2(x)} \frac{\mathbf{y}^0}{\sqrt{|\mathcal{N}_2(x)||\mathcal{N}_2(y)|}} \\ &+ \alpha \sum_{b \in \mathcal{N}_1(u)} \frac{(\mathbf{b}^0)'}{\sqrt{|\mathcal{N}_1(u)||\mathcal{N}_1(b)|}} \sum_{c \in \mathcal{N}_1(a)} \frac{\mathbf{c}^0}{\sqrt{|\mathcal{N}_1(a)||\mathcal{N}_1(c)|}} \\ &+ \sqrt{\alpha(1-\alpha)} \sum_{b \in \mathcal{N}_1(u)} \frac{(\mathbf{b}^0)'}{\sqrt{|\mathcal{N}_1(u)||\mathcal{N}_1(b)|}} \mathbf{x}^0\end{aligned}$$

$$\begin{aligned}\mathbf{S}(a, x) = &2\alpha \sum_{b \in \mathcal{N}_1(a)} \frac{(\mathbf{b}^0)'}{\sqrt{|\mathcal{N}_1(a)||\mathcal{N}_1(b)|}} \sum_{y \in \mathcal{N}_2(x)} \frac{\mathbf{y}^0}{\sqrt{|\mathcal{N}_2(x)||\mathcal{N}_2(y)|}} \\ &+ \frac{\alpha}{|\mathcal{N}_1(a)|} \sum_{b \in \{\mathcal{N}_1(a) \cap \mathcal{L}_1\}} \frac{1}{|\mathcal{N}_1(b)|} \\ &+ \frac{\alpha}{|\mathcal{N}_2(x)|} \sum_{y \in \{\mathcal{N}_2(x) \cap \mathcal{L}_2\}} \frac{1}{|\mathcal{N}_2(x)|} + (1 - \alpha)\end{aligned}$$

Note $\mathbf{S}(a, v)$ is omitted as it is similar to $\mathbf{S}(u, x)$. We denote $\mathbf{S}_1(u, a) = [\sum_{b \in \mathcal{N}_1(u)} \frac{\mathbf{b}^0}{\sqrt{|\mathcal{N}_1(u)||\mathcal{N}_1(b)|}}]'[\sum_{c \in \mathcal{N}_1(a)} \frac{\mathbf{c}^0}{\sqrt{|\mathcal{N}_1(a)||\mathcal{N}_1(c)|}}]$, $\mathbf{S}_2(x, v) = [\sum_{y \in \mathcal{N}_2(x)} \frac{\mathbf{y}^0}{\sqrt{|\mathcal{N}_2(y)||\mathcal{N}_2(x)|}}]'[\sum_{z \in \mathcal{N}_2(v)} \frac{\mathbf{z}^0}{\sqrt{|\mathcal{N}_2(z)||\mathcal{N}_1(v)|}}]$ which measure the weighted number of common neighboring anchor nodes. Recall Eq. (4), $\mathbf{L}(u, v) = \mathbf{L}(u, x) = 0$ and $\mathbf{L}(a, x) = 1$, then we have

$$\mathbf{S}(u, v) = \alpha\tilde{\mathbf{A}}_1(a, :)\mathbf{L}\tilde{\mathbf{A}}_2(:, v) + (1 - \alpha)\mathbf{L}(u, v)$$

$$\begin{aligned}\mathbf{S}(u, x) = &\alpha\tilde{\mathbf{A}}_1(u, :)\mathbf{L}\tilde{\mathbf{A}}_2(:, x) + (1 - \alpha)\mathbf{L}(u, x) + \alpha\mathbf{S}_1(u, a) \\ &+ \sqrt{\alpha(1-\alpha)} \frac{\mathbf{A}_1(u, a)}{\sqrt{|\mathcal{N}_1(u)||\mathcal{N}_1(a)|}}\end{aligned}$$

$$\mathbf{S}(a, x) = 2\alpha\tilde{\mathbf{A}}_1(a, :)\mathbf{L}\tilde{\mathbf{A}}_2(:, x) + \alpha(\mathbf{S}_1(a, a) + \mathbf{S}_2(x, x)) + (1 - \alpha)\mathbf{L}(a, x)$$

As we can see, the alignments based on the embeddings learned by Eq. (5) in the *first* iteration are equivalent to the semi-supervised *FINAL* (i.e., Eq. (3)) with $\mathbf{S}^0 = \mathbf{L}$ except the additional intra-network proximity (e.g., $\mathbf{S}_1, \mathbf{S}_2$) and scaling terms. This completes the proof. □

This lemma implies that we can design a special relational graph convolutional network (RelGCN) to encode the alignment consistency. Specifically, we formulate the t-th RelGCN layer as

$$\begin{aligned}\mathbf{u}^t &= \sqrt{\alpha} \sum_{b \in \mathcal{N}_1(u)} \frac{\mathbf{W}_1^t \mathbf{b}^{t-1}}{\sqrt{|\mathcal{N}_1(u)||\mathcal{N}_1(b)|}} + \sqrt{1-\alpha}\mathbf{W}_0^t \mathbf{u}^{t-1} \\ \mathbf{v}^t &= \sqrt{\alpha} \sum_{y \in \mathcal{N}_2(v)} \frac{\mathbf{W}_2^t \mathbf{y}^{t-1}}{\sqrt{|\mathcal{N}_2(v)||\mathcal{N}_2(y)|}} + \sqrt{1-\alpha}\mathbf{W}_0^t \mathbf{v}^{t-1} \\ \mathbf{a}^t = \mathbf{x}^t &= \sqrt{\alpha} \sum_{b \in \mathcal{N}_1(a)} \frac{\mathbf{W}_1^t \mathbf{b}^{t-1}}{\sqrt{|\mathcal{N}_1(a)||\mathcal{N}_1(b)|}} + \sqrt{1-\alpha}\mathbf{W}_0^t \mathbf{x}^{t-1} \\ &+ \sqrt{\alpha} \sum_{y \in \mathcal{N}_2(x))} \frac{\mathbf{W}_2^t \mathbf{y}^{t-1}}{\sqrt{|\mathcal{N}_2(x)||\mathcal{N}_2(y)|}}\end{aligned} \tag{6}$$

where $\mathbf{W}_0^t, \mathbf{W}_1^t, \mathbf{W}_2^t \in \mathbb{R}^{d_t \times d_{t-1}}$ and d_t denotes the embedding dimension in the t-th layer of RelGCN. Note the slight differences between RelGCN and the existing R-GCN [20] are the normalization terms and additional scaling terms. In addition, we name the RelGCN layer without weight parameters (i.e., Eq. (5)) as RelGCN-U.

We then design the whole model architecture shown in Figure 2. The key idea is to leverage RelGCNs to learn node embeddings that describe the *relative 'positions'* of the nodes w.r.t. the anchor nodes [27], followed by a linear layer to learn the final output embeddings. In particular, given the prior alignment matrix $\mathbf{L}$, we first decompose it into two rank-L matrices by $\mathbf{L} = \mathbf{L}_1^T\mathbf{L}_2$ as the initial embeddings. Then we feed them into RelGCN-U to incorporate the alignment consistency. However, with one RelGCN-U layer, only those nodes that are connected with anchor nodes can be reached by propagation from anchor nodes, while other distant non-anchor nodes cannot be reached. In this way, we apply random walk with restart [24] to measure the proximities of non-anchor nodes w.r.t. the anchor nodes as the initial relative positions, i.e.,

$$\mathbf{r}_{i1} = (1 - p)\hat{\mathbf{A}}_1\mathbf{r}_{i1} + p\hat{\mathbf{e}}_{i1},\ \mathbf{r}_{i2} = (1 - p)\hat{\mathbf{A}}_2\mathbf{r}_{i2} + p\hat{\mathbf{e}}_{i2}$$

where the restart probability p is set to 0.85 following the classic choice, and $\hat{\mathbf{A}}_1, \hat{\mathbf{A}}_2$ are the normalized matrices of $\mathbf{A}_1, \mathbf{A}_2$. The restart vectors $\hat{\mathbf{e}}_{i1} \in \mathbb{R}^{n_1}, \hat{\mathbf{e}}_{i2} \in \mathbb{R}^{n_2}$ only have one nonzero value at $\hat{\mathbf{e}}_{i1}(a) = 1$ and $\hat{\mathbf{e}}_{i2}(x) = 1$. After achieving the stationary distributions, we set $\mathbf{u}^0 = [\mathbf{r}_{11}(u), \mathbf{r}_{21}(u), \cdots, \mathbf{r}_{L1}(u)]'$ for non-anchor node-u in $\mathcal{G}_1$, and similarly for non-anchor nodes v in $\mathcal{G}_2$. We denote the output embeddings by RelGCN-U by, say, $\mathbf{a}^1, \mathbf{x}^1, \mathbf{u}^1, \mathbf{v}^1$.

However, the alignment scores among the close neighborhood might be too consistent with each other to distinguish the precise node alignments due to the over-smoothness (e.g., alignments of (c, f) and (c, g) in Figure 2). To mitigate the issue, we leverage RelGCN with attention mechanism to rescale the relative positions. Mathematically, the attention coefficients are computed by

$$\begin{aligned}\hat{\mathbf{u}} &= \sqrt{\alpha} \sum_{b \in \mathcal{N}_1(u)} \frac{\mathbf{W}_1\mathbf{X}_1(:, b)}{\sqrt{|\mathcal{N}_1(u)||\mathcal{N}_1(b)|}} + \sqrt{1-\alpha}\mathbf{W}_0\mathbf{X}_1(:, u) \\ \hat{\mathbf{a}} = \hat{\mathbf{x}} &= \sqrt{\alpha} \sum_{b \in \mathcal{N}_1(a)} \frac{\mathbf{W}_1\mathbf{X}_1(:, b)}{\sqrt{|\mathcal{N}_1(a)||\mathcal{N}_1(b)|}} + \sqrt{1-\alpha}\mathbf{W}_0\mathbf{X}_1(:, a) \\ &+ \sqrt{\alpha} \sum_{y \in \mathcal{N}_2(x)} \frac{\mathbf{W}_2\mathbf{X}_2(:, y)}{\sqrt{|\mathcal{N}_2(x)||\mathcal{N}_2(y)|}} \\ c_{ua} &= \frac{\exp(\mathbf{w}_c'[\hat{\mathbf{u}}\|\hat{\mathbf{a}}])}{\sum_{b \in \mathcal{L}_1} \exp(\mathbf{w}_c'[\hat{\mathbf{u}}\|\hat{\mathbf{b}}])}\end{aligned}$$

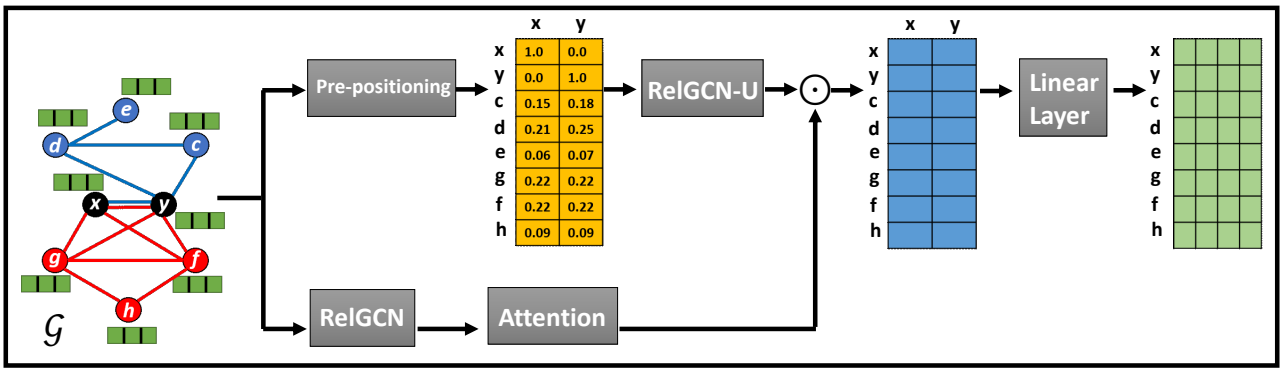


Figure 2: Overall architecture of NeXtAlign.

and similarly for computing c_{vx} where $\mathbf{w}_c$ is a parameter vector. We scale the relative positions by

$$\tilde{\mathbf{u}} = \hat{\mathbf{u}} \odot \mathbf{c}_u,\ \tilde{\mathbf{v}} = \hat{\mathbf{v}} \odot \mathbf{c}_v,\ \tilde{\mathbf{a}} = \tilde{\mathbf{x}} = \hat{\mathbf{x}} \odot \mathbf{c}_x \tag{7}$$

where $\mathbf{c}_u(i) = c_{ua}, \mathbf{c}_v(i) = c_{vx}$. To learn the final output node embeddings, we apply a simple linear layer, i.e.,

$$\mathbf{u} = \mathbf{W}\tilde{\mathbf{u}},\ \mathbf{v} = \mathbf{W}\tilde{\mathbf{v}},\ \mathbf{a} = \mathbf{x} = \mathbf{W}\tilde{\mathbf{x}} \tag{8}$$

where $\mathbf{W} \in \mathbb{R}^{d \times |\mathcal{L}|}$ is the corresponding weight parameter matrix.

3.3 Model Training

To learn the node embeddings, we first consider the following loss functions on anchor links that capture both intra-network information by J_a, J_x and inter-network information by J_{ax}.

$$J_a = -\sum_b \left[p_d(b|a)\log\sigma(\mathbf{b}'\mathbf{a}) + kp_n(b|a)\log\sigma(-\mathbf{b}'\mathbf{a})\right]$$

$$J_x = -\sum_y \left[p_d(y|x)\log\sigma(\mathbf{y}'\mathbf{x}) + kp_n(y|x)\log\sigma(-\mathbf{y}'\mathbf{x})\right]$$

$$J_{ax} = -\sum_b \left[p_{dc}(b|x)\log\sigma(\mathbf{b}'\mathbf{x}) + kp_{nc}(b|x)\log\sigma(-\mathbf{b}'\mathbf{x})\right] - \sum_y \left[p_{dc}(y|a)\log\sigma(\mathbf{y}'\mathbf{a}) + kp_{nc}(y|a)\log\sigma(-\mathbf{y}'\mathbf{a})\right]$$

$$J = \sum_{(a,x)\in\mathcal{L}} J_{(a,x)} = \sum_{(a,x)\in\mathcal{L}} J_a + J_x + J_{ax}$$

where $\sigma(\cdot)$ is the sigmoid function. The probability distributions p_d, p_n sample the positive and negative context node pairs within the same network respectively, while p_{dc}, p_{nc} sample positive and negative alignment pairs across different networks. Note that in the above loss functions, we assume for simplicity that the probabilities for the same goal are calculated by the same function (e.g., $p_d(\cdot|a)$ in $\mathcal{G}_1$ vs. $p_d(\cdot|x)$ in $\mathcal{G}_2$ using the same function but different inputs) and the numbers of negative samples by p_n, p_{nc} are same (i.e., k). Since $\mathbf{a} = \mathbf{x},\ \forall(a,x) \in \mathcal{L}$, we can rewrite the loss related to the anchor link (a,x) as below.

$$J_{(a,x)} = -\sum_b \left[[p_d(b|a) + p_{dc}(b|x)]\log\sigma(\mathbf{b}'\mathbf{x}) + k[p_n(b|a) + p_{nc}(b|x)]\log\sigma(-\mathbf{b}'\mathbf{x})\right] \tag{9}$$

$$- \sum_y \left[[p_d(y|x) + p_{dc}(y|a)]\log\sigma(\mathbf{y}'\mathbf{x}) + k[p_n(y|x) + p_{nc}(y|a)]\log\sigma(-\mathbf{y}'\mathbf{x})\right] \tag{10}$$

For the anchor link (a,x), we derive the conditions of the optimal node embedding that minimize $J_{(a,x)}$ inspired by [28].

Lemma 2. *Given an anchor link (a,x), the optimal embeddings that minimize $J_{(a,x)}$ satisfy for non-anchor nodes $b \in \bar{\mathcal{L}}_1, y \in \bar{\mathcal{L}}_2$,*

$$\mathbf{b}'\mathbf{x} = -\log\frac{kp_n(b|a) + kp_{nc}(b|x)}{p_d(b|a) + p_{dc}(b|x)} \tag{11}$$

$$\mathbf{y}'\mathbf{x} = -\log\frac{kp_n(y|x) + kp_{nc}(y|a)}{p_d(y|x) + p_{dc}(y|a)} \tag{12}$$

and for anchor nodes such that $(b,y) \in \mathcal{L}$,

$$\mathbf{b}'\mathbf{x} = \mathbf{y}'\mathbf{x} = -\log\frac{k[p_n(b|a) + p_{nc}(b|x) + p_n(y|x) + p_{nc}(y|a)]}{p_d(b|a) + p_{dc}(b|x) + p_d(y|x) + p_{nc}(y|a)} \tag{13}$$

Proof. See Appendix. □

However, the above lemma requires sufficient sampled node pairs, while we only have a limited number of samples in the real case. In this way, we further consider to minimize the empirical risk for an anchor link (a,x) as

$$J^B_{(a,x)} = -\frac{1}{B}\sum_{i_1,i_2,j_1,j_2} \log\sigma(\mathbf{b}'_{i_1}\mathbf{x}) + \log\sigma(\mathbf{b}'_{i_2}\mathbf{x}) + \log\sigma(\mathbf{y}'_{j_1}\mathbf{x}) + \log\sigma(\mathbf{y}'_{j_2}\mathbf{x}) - \frac{1}{B}\sum_{i_3,i_4,j_3,j_4}\Big[\log\sigma(-\mathbf{b}'_{i_3}\mathbf{x}) + \log\sigma(-\mathbf{b}'_{i_4}\mathbf{x}) + \log\sigma(-\mathbf{y}'_{j_3}\mathbf{x}) + \log\sigma(-\mathbf{y}'_{j_4}\mathbf{x})\Big]$$

where B is the number of positive samples and accordingly kB is the size of negative samples. In addition, (1) b_{i_1}, y_{j_1} are sampled from $p_d(\cdot|a), p_d(\cdot|x)$, (2) b_{i_2}, y_{j_2} are sampled from $p_{dc}(\cdot|x), p_{dc}(\cdot|a)$, (3) b_{i_3}, y_{j_3} are sampled from $p_n(\cdot|a), p_n(\cdot|x)$, and (4) b_{i_4}, y_{j_4} are sampled from $p_{nc}(\cdot|x), p_{nc}(\cdot|a)$ respectively. Furthermore, by defining $\boldsymbol{\theta} = [\mathbf{b}'_1\mathbf{x}, \cdots, \mathbf{b}'_{n_1}\mathbf{x}, \mathbf{y}'_1\mathbf{x}, \cdots, \mathbf{y}'_{n_2}\mathbf{x}]$, we denote $\boldsymbol{\theta}^*$ as the optimal solution to $J_{(a,x)}$ and $\boldsymbol{\theta}^B$ similarly for the empirical risk $J^B_{(a,x)}$. Then we can derive the mean square error in the following lemma.

Lemma 3. *Denote $\Delta\boldsymbol{\theta}_b = \boldsymbol{\theta}^B_b - \boldsymbol{\theta}^*_b$ and $\Delta\boldsymbol{\theta}_y = \boldsymbol{\theta}^B_y - \boldsymbol{\theta}^*_y$. The mean square errors for nodes $b \in \bar{\mathcal{L}}_1$ and $y \in \bar{\mathcal{L}}_2$ can be formulated by*

$$\mathbb{E}\left[\Delta\theta^2_b\right] = \frac{1}{B}\left[\frac{1}{p_d(b|a) + p_{dc}(b|x)} + \frac{1}{kp_n(b|a) + kp_{nc}(b|x)} - C\right]$$

$$\mathbb{E}[\Delta\theta^2_y] = \frac{1}{B}\left[\frac{1}{p_d(y|x) + p_{dc}(y|a)} + \frac{1}{kp_n(y|x) + kp_{nc}(y|a)} - C\right] \tag{14}$$

For nodes $b \in \mathcal{L}_1$ and $y \in \mathcal{L}_2$, the mean square error is computed by

$$\mathbb{E}\left[\Delta\theta^2_b\right] = \mathbb{E}[\Delta\theta^2_y] = \frac{1}{B}\left[\frac{1}{p_1} + \frac{1}{kp_2} - C\right] \tag{15}$$

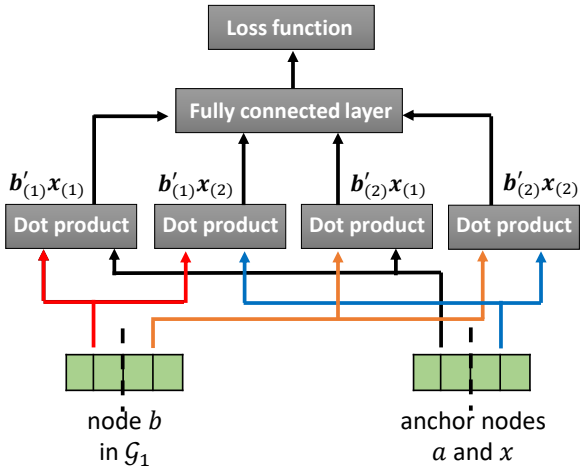


Figure 3: Illustration of embedding interactions.

where $C = 1 + \frac{1}{k}$, $p_1 = p_d(b|a) + p_{dc}(b|x) + p_d(y|x) + p_{dc}(y|a)$ *and* $p_2 = p_n(b|a) + p_{nc}(b|x) + p_n(y|x) + p_{nc}(y|a)$.

Proof. See Appendix. □

Given the above lemma, the question now comes to *how to design these distributions*. For a single network where only p_d and p_n are considered, [28] proposes that p_n is positively correlated to p_d. And if node-b has a high embedding similarity with node-a, it is likely to be a negative sample. This can be considered as a *hard* negative sample which in the task of recommendation could separate the negative items from positive ones for a certain user. However, in pairwise network alignment where we have two input networks, different sampling distributions serve different purposes, which as we show in the following may lead to the competing designs. First (for p_d), a typical goal of p_d is to sample nodes that are similar to the center nodes such that the sampled nodes are likely to co-occur with the center nodes in some manually extracted contexts [8, 16]. Second (for p_n), we follow the intuition in unsupervised network embedding that close neighbors should be similar while distant nodes should be dissimilar in terms of embedding vectors [14]. This implies that distant/dissimilar nodes are more likely to be sampled by p_n. Third (for p_{dc}), we use it to sample positive alignment pairs across networks that are likely to form the alignments and preserve the *alignment consistency* similar to Eq. (1). This implies p_{dc} should be positively correlated to the embedding similarities. Fourth (for p_{nc}), we first note that network alignment can be considered as a special recommendation task where the anchor link of two nodes is analogized as the *only* positive item for a user. In this way, we would like to use p_{nc} to provide *hard* negative alignment pairs as in recommendation [18, 28]. That is, nodes in $\mathcal{G}_1$ that currently have high embedding similarities with anchor node-x are likely to be the hard negative samples. By doing so, we could attain the *alignment disparity*. That is, the embeddings of the nodes that are likely to mislead the alignments are encouraged to be more separable from node-x. We remark that while selecting nodes as the negative alignment pairs, they are supposed not to violate the overall alignment consistency. In addition, given Lemma 3 and the design principles of p_d, p_n, p_{dc}, in order to estimate high-probability node pairs (i.e., high p_d, p_{dc} and low p_n) more accurately, we need p_{nc} to be large which coincides with the designed positive correlation.

But with these designs, nodes that have high embedding similarities to anchor nodes are likely to be sampled as both positive context/alignment pairs and negative alignment pairs. In other words, suppose node-b is sampled to form a positive context pair with anchor node-a and a negative alignment pair with the anchor node-x simultaneously. Then according to Eq. (9), it means that the node pair (b, x) should be classified into both the positive class and the negative class, i.e., two competing objectives. To address this issue, instead of simply using $\mathbf{b}'\mathbf{x}$ to compute p_d, p_n, p_{dc}, p_{nc}, we divide the node embedding vectors to two parts, i.e., $\mathbf{b} = [\mathbf{b}_{(1)} \| \mathbf{b}_{(2)}]$ and $\mathbf{x} = [\mathbf{x}_{(1)} \| \mathbf{x}_{(2)}]$ where $\mathbf{b}_{(1)}, \mathbf{b}_{(2)}, \mathbf{x}_{(1)}, \mathbf{x}_{(2)} \in \mathbb{R}^{d/2}$. Each part of the vector aims to capture different information. Take $\mathbf{b}$ of node-b in $\mathcal{G}_1$ as an example. Here, $\mathbf{b}_{(1)}$ captures the local neighborhood information of node-b in $\mathcal{G}_1$ while $\mathbf{b}_{(2)}$ encodes how node-b posits in the context of $\mathcal{G}_2$. Then we are able to design a new alignment scoring function that allows the interactions among different parts of embeddings as shown in Figure 3. Specifically, we define

$$\mathbf{b} \star \mathbf{x} = w_1 \mathbf{b}'_{(1)} \mathbf{x}_{(1)} + w_2 \mathbf{b}'_{(1)} \mathbf{x}_{(2)} + w_3 \mathbf{b}'_{(2)} \mathbf{x}_{(1)} + w_4 \mathbf{b}'_{(2)} \mathbf{x}_{(2)} \quad (16)$$

where $[w_1, w_2, w_3, w_4]$ is a vector of parameters to be learned that measure the importance of different terms. By replacing the original inner products in $J^B_{(a,x)}$ with Eq. (16), the embedding similarities are determined by different aspects. In particular, since $\mathbf{a} = \mathbf{x}$, the first term $\mathbf{b}'_{(1)}\mathbf{x}_{(1)} = \mathbf{b}'_{(1)}\mathbf{a}_{(1)}$ implies the intra-network proximity between node-b in $\mathcal{G}_1$ and anchor node-x and hence can be used in the sampling probabilities $p_d(b|a)$ and $p_n(b|a)$. Second, the last term $\mathbf{b}'_{(2)}\mathbf{x}_{(1)}$ describes how likely, in the context of $\mathcal{G}_2$, node-b interacts with anchor node-x, and thus can be used to measure to what extent they are aligned. In this way, this term can be used in the sampling distribution $p_{dc}(b|x)$. Lastly, the middle two terms capture how likely that two nodes b and x are interacted in a way similar as in recommendation. For example, the term $\mathbf{b}'_{(1)}\mathbf{x}_{(2)}$ can be considered as the way we do inner products in social recommendation as $\mathbf{b}_{(1)}$ and $\mathbf{x}_{(2)}$ are the embeddings of two nodes in the context of their own networks. This allows us to use this two terms to formulate the cross-network negative sampling distribution $p_{nc}(b|x)$ to provide hard negative samples. Consequently, the probability distributions for $\mathcal{G}_1$ can be formulated as

$$p_d(b|a) = \frac{\sigma(\mathbf{b}'_{(1)}\mathbf{x}_{(1)})}{\sum_{c\in\mathcal{V}_1}\sigma(\mathbf{c}'_{(1)}\mathbf{x}_{(1)})} \quad (17)$$

$$p_n(b|a) = \frac{\sigma(-\mathbf{b}'_{(1)}\mathbf{x}_{(1)})}{\sum_{c\in\mathcal{V}_1}\sigma(-\mathbf{c}'_{(1)}\mathbf{x}_{(1)})} \quad (18)$$

$$p_{dc}(b|x) = \frac{\sigma(\mathbf{b}'_{(2)}\mathbf{x}_{(2)})}{\sum_{c\in\mathcal{V}_1}\sigma(\mathbf{c}'_{(2)}\mathbf{x}_{(2)})} \quad (19)$$

$$p_{nc}(b|x) = \frac{\sigma(\mathbf{b}'_{(1)}\mathbf{x}_{(2)} + \mathbf{b}'_{(2)}\mathbf{x}_{(1)})}{\sum_{c\in\mathcal{V}_1}\sigma(\mathbf{c}'_{(1)}\mathbf{x}_{(2)} + \mathbf{c}'_{(2)}\mathbf{x}_{(1)})} \quad (20)$$

and for $\mathcal{G}_2$ they can be computed similarly. In this way, we can encode different aspects of the sampling design principles simultaneously and strike a balance among them through the learning process. Since the real p_d is often unknown and we define its approximation by *node2vec* [8] to explicitly provide positive context pairs. In addition, as aforementioned, the true positive alignment

pair for an anchor node is the anchor link itself. Thus the nodes sampled by $p_{dc}(b|x)$ can only be considered to form the *intermediate* positive alignment pairs. In this way, we add another loss term in $J^B_{(a,x)}$ to encode the differences between the anchor links and intermediate positive alignment pairs, which gives the final loss

$$
\begin{aligned}
J^B_{(a,x)} = & \frac{1}{B}\sum_{i_1}\log\sigma(\mathbf{b}_{i_1}\star\mathbf{x}) + \frac{1}{B}\sum_{j_1}\log\sigma(\mathbf{y}_{j_1}\star\mathbf{x}) \\
& + \frac{1}{B}\sum_{i_2}\log\sigma(\mathbf{b}_{i_2}\star\mathbf{x}) + \max\{0, \sigma(\mathbf{b}_{i_2}\star\mathbf{x}) - \sigma(\mathbf{x}\star\mathbf{x}) + \lambda\} \\
& + \frac{1}{B}\sum_{j_2}\log\sigma(\mathbf{y}_{j_2}\star\mathbf{x}) + \max(0, \sigma(\mathbf{y}_{j_2}\star\mathbf{x}) - \sigma(\mathbf{x}\star\mathbf{x}) + \lambda) \\
& + \frac{1}{B}\sum_{i_3,i_4}\log\sigma(-\mathbf{b}_{i_3}\star\mathbf{x}) + \log\sigma(-\mathbf{b}_{i_4}\star\mathbf{x}) \\
& + \frac{1}{B}\sum_{j_3,j_4}\log\sigma(-\mathbf{y}_{j_3}\star\mathbf{x}) + \log\sigma(-\mathbf{y}_{j_4}\star\mathbf{x})
\end{aligned}
$$

where λ is the margin.

4 EXPERIMENTAL RESULTS

We evaluate the proposed NeXtAlign in the following aspects:

- Q1. How accurate is NeXtAlign for network alignment?
- Q2. To what extent does the proposed method benefit from different components of the model?

4.1 Experimental Setup

The statistics of the datasets are summarized in Table 2. Detailed descriptions and settings are introduced in Appendix[1].

4.2 Effectiveness Results

Alignment without node attributes. We first evaluate the alignment performance without using node attributes under different training ratios. The results of the experiments using 20% and 10% training data are summarized in Table 3 and Table 4 respectively. We have the following observations. First, by comparing with the consistency-based semi-supervised FINAL, our proposed method achieves an up to 20% improvement in both Hits@30 and Hits@10, which indicates that despite their close relationships in capturing the alignment consistency, our proposed method benefits from encompassing alignment disparity. Second, our proposed method consistently outperforms all the other embedding based alignment methods (i.e., Bright, NetTrans, IONE and CrossMNA). In particular, our method achieves an at least 3% improvement in Hits@30 compared to the best competitor. This demonstrates that our method can learn more meaningful node embeddings for the task of network alignment. Third, in the scenarios *S2* and *S3* where networks to be aligned are disparate with each other in terms network structure (e.g., significant differences in edge density), our proposed method achieves more improvements over the baseline methods than in the scenario *S1*. This implies that our method can perform better to align networks where alignment consistency might not be much helpful. Lastly, even with fewer training data (i.e., 10% training data), our method still outperforms other baseline methods.

Alignment with node attributes. Moreover, we evaluate the performance of node attributed network alignment in *S1*. The results are shown in Table 5. As we can see, all the methods benefit a lot from leveraging node attributes to infer more accurate node alignments. In the meanwhile, our method still outperforms all the other baseline methods under different training ratios.

[1]The code can be found at https://github.com/sizhang92/NextAlign-KDD21.

Table 2: Data statistics.

Scenarios	Networks	# of nodes	# of edges	# of attributes
S1	ACM	9,872	39,561	17
	DBLP	9,916	44,808	17
S2	Foursquare	5,313	54,233	0
	Twitter	5,120	130,575	0
S3	Phone	1,000	41,191	0
	Email	1,003	4,627	0

4.3 Analysis of the Method

In this subsection, we conduct several ablation studies to validate different components of the proposed method.

Ablation study on model design. We compare our proposed method with the following variants: (1) RWR, which uses initial embeddings with pre-positioning by random walk with restart (e.g., $\mathbf{u}^0, \mathbf{v}^0$), (2) RelGCN-U, which uses the output embedding by RelGCN-U layer as the output node embeddings, and (3) RelGCN-C, which uses the re-scaled relative positions as the final embeddings (e.g., $\tilde{\mathbf{u}}$). The results are shown in Figure 4. As we can see, the proposed method NeXtAlign performs the best, validating the necessities of all components in the whole model.

Ablation study on sampling strategies. To demonstrate that our proposed sampling method is indeed beneficial, we compare our proposed method with different variants by changing the sampling strategies. Specifically, we compare with the model variants that for an anchor node-x, (1 - uniform) uniformly at random sample negative context pairs and alignment pairs, (2 - degree) sample negative pairs based on the node degree (e.g., $p_n(v|x) \propto d_v^{3/4}$ and $p_{nc}(u|x) \propto d_u^{3/4}$), and (3 - positive) sample nodes by the distribution which is positively correlated to the inner product among node embeddings (e.g., $p_n(v|x) \propto \sigma(\mathbf{v}'\mathbf{x})$ and $p_{nc}(u|x) \propto \sigma(\mathbf{u}'\mathbf{x})$). The results are summarized in Table 6. We observe that the pre-defined sampling strategies perform the worst. While using positively correlation helps improve the alignment performance against the pre-defined distributions, our proposed sampling strategy still performs the best. This validates the benefits of using both negative correlation and positive correlation.

Ablation study on Eq. (16)**.** Here, we compare the alignment performance with the model variant by replacing the scoring function Eq. (16) with the simple inner product using 20% training data. As we can see in Figure 5, using Eq. (16) indeed boosts the performance.

Parameter study on the number of negative samples. Moreover, we analyze how alignment performance varies with different number of negative samples $k = [1, 5, 10, 20, 50, 100, 500]$ on different datasets. The comparisons are shown in Figure 6. As we can see, the alignment performance is stable under different settings of sampling size k. In addition, using a relative small size of negative samples (i.e., $k \in [5, 20]$) achieves a good overall performance.

5 RELATED WORKS

Network alignment. Many network alignment methods explicitly formulate the alignment consistency into various optimization problems. A classic formulation of network alignment is to minimize

Table 3: Results with 20% training data.

	ACM-DBLP		Foursquare-Twitter		Phone-Email	
	Hits@10	Hits@30	Hits@10	Hits@30	Hits@10	Hits@30
NeXtAlign	**0.8417±0.0032**	**0.9011±0.0081**	**0.2956±0.0096**	**0.4174±0.0066**	**0.3926±0.0168**	**0.6748±0.0105**
Bright	0.7904±0.0041	0.8669±0.0041	0.2500±0.0154	0.3206±0.0097	0.2570±0.0091	0.5344±0.0086
NetTrans	0.7925±0.0065	0.8356±0.0082	0.2468±0.0036	0.3458±0.0098	0.2650±0.0025	0.5325±0.0075
FINAL	0.6768±0.0080	0.8237±0.0098	0.2357±0.0091	0.3457±0.0091	0.2203±0.0151	0.4586±0.0184
IONE	0.7476±0.0125	0.8453±0.0097	0.1624±0.0109	0.2918±0.0209	0.3779±0.0131	0.6444±0.0084
CrossMNA	0.6532±0.0042	0.7900±0.0041	0.0236±0.0172	0.0751±0.0384	0.1542±0.0041	0.4045±0.0115

Table 4: Results with 10% training data.

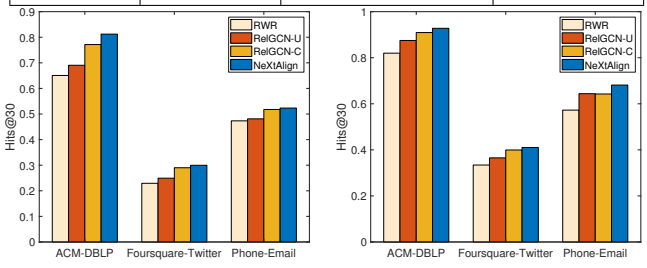
	ACM-DBLP		Foursquare-Twitter		Phone-Email	
	Hits@10	Hits@30	Hits@10	Hits@30	Hits@10	Hits@30
NeXtAlign	**0.7242±0.0035**	**0.8156±0.0018**	**0.1946±0.0115**	**0.2969±0.0149**	**0.2785±0.0165**	**0.5446±0.0185**
Bright	0.7022±0.0068	0.7640±0.0068	0.1705±0.0034	0.2454±0.0096	0.2027±0.0133	0.4530±0.0099
NetTrans	0.6385±0.0076	0.7402±0.0099	0.1581±0.0021	0.2410±0.0076	0.1767±0.0085	0.4089±0.0105
FINAL	0.4602±0.0092	0.6490±0.0074	0.1387±0.0087	0.2371±0.0124	0.1749±0.0124	0.3857±0.0158
IONE	0.4773±0.0181	0.6108±0.0168	0.069±0.0142	0.1670±0.0219	0.1802±0.0087	0.4440±0.0093
CrossMNA	0.3690±0.0071	0.5063±0.0065	0.0242±0.0038	0.0775±0.0085	0.1139±0.0085	0.3506±0.0132

Table 5: Alignment on ACM-DBLP with attributes.

	10% training data		20% training data	
	Hits@10	Hits@30	Hits@10	Hits@30
NeXtAlign	**0.785±0.010**	**0.871±0.009**	**0.872±0.016**	**0.942±0.003**
Bright	0.781±0.004	0.862±0.003	0.797±0.004	0.870±0.006
NetTrans	0.708±0.004	0.846±0.009	0.841±0.010	0.916±0.013
FINAL	0.651±0.013	0.817±0.009	0.825±0.008	0.916±0.006

Table 6: Ablations study on sampling strategies by Hits@30.

	ACM-DBLP	Foursquare-Twitter	Phone-Email
NeXtAlign	0.9277	0.4103	0.6813
Uniform	0.8975	0.3924	0.6525
Degree	0.9093	0.3923	0.6637
Positive	0.9097	0.4040	0.6650



(a) 10% training data. **(b) 20% training data.**

Figure 4: Ablation study on model architecture.

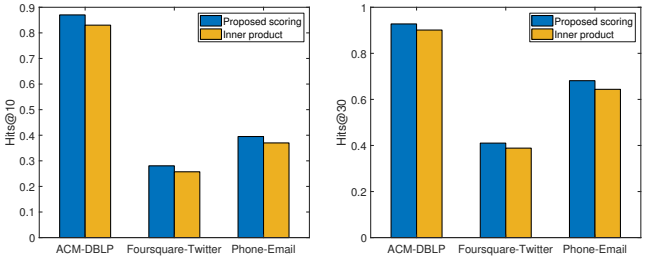
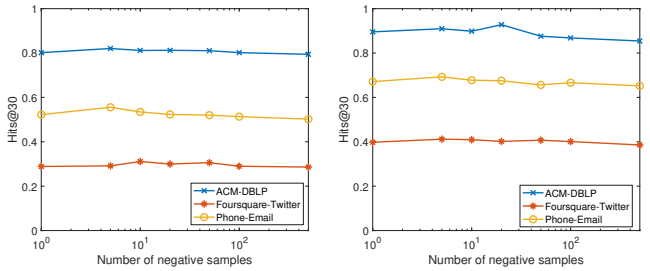


Figure 5: Ablation study on the scoring function.

$\|\mathbf{A}_2 - \mathbf{P}'\mathbf{A}_1\mathbf{P}\|_F^2$ where $\mathbf{P}$ is constrained to be a permutation matrix. However, this leads to a NP-hard problem. To solve it, a variety of approximation methods have been proposed, including orthogonal relaxation [5], sparse probabilistic relaxation [31] and doubly stochastic relaxation [10]. Moreover, the alignment consistency can be also formulated into maximizing the number of neighboring node pairs that are aligned [2]. Zhang et al. interpret the classic random walk based method IsoRank [21] as an optimization problem that minimizes the alignment differences of neighboring node pairs [32]. Additionally, there exist many methods that implicitly maximize the alignment consistency. For example, MAGNA aims to optimize the edge conservation by a generic algorithm [19] and MAGNA++ encodes both node and edge conservation [25]. However, alignment consistency itself might over-smoothen the alignments.



(a) 10% training data. **(b) 20% training data.**

Figure 6: Hits@30 with different sizes of negative samples.

Many embedding based methods have been proposed to infer node alignments by learning node embeddings [3, 9, 13, 27, 37]. Many of them can more or less mitigate the over-smoothness by sampling negative alignment pairs. However, their negative sampling distributions are mostly pre-defined, which might be insufficient to correctly encourage the alignment disparity. Besides, in the unsupervised setting, adversarial learning can be used to align nodes by matching node embedding distributions [4, 17].

Negative sampling. Negative sampling has been widely used in many tasks such as word embedding [15], recommendation [6, 29, 38] and network embedding [16, 28, 36]. For network embedding, one common choice is to sample negative nodes based on the

node degree [8, 16]. More recently, Yang et al. systematically study negative sampling in network embedding and propose the negative sampling distribution should be positively and sub-linearly correlated to the positive sampling distribution [28]. In the meanwhile, some other works advocate a competing design to sample distant nodes that are expected to be dissimilar to the center nodes [1, 14].

6 CONCLUSION

In this paper, we study the semi-supervised network alignment problem. Different from the existing works, we aim to encompass both alignment consistency and alignment disparity. Specifically, we unveil that the node embeddings learned by the special graph convolutional network RelGCN-U can infer node alignments that preserve alignment consistency. Furthermore, we theoretically analyze the expected loss and empirical loss, which motivates the designs of sampling distributions in network alignment. Based on these, we propose a novel network alignment method NeXtAlign that achieves a good trade-off between alignment consistency and alignment disparity via the learning process. Extensive experiments show that the proposed method significantly outperforms all the baseline methods.

7 ACKNOWLEDGEMENT

This work is supported by National Science Foundation under grant No. 1947135. The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

[1] Mohammadreza Armandpour, Patrick Ding, Jianhua Huang, and Xia Hu. 2019. Robust negative sampling for network embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3191–3198.

[2] Mohsen Bayati, Margot Gerritsen, David F Gleich, Amin Saberi, and Ying Wang. 2009. Algorithms for large, sparse network alignment problems. In *2009 Ninth IEEE International Conference on Data Mining*. IEEE, 705–710.

[3] Xiaokai Chu, Xinxin Fan, Di Yao, Zhihua Zhu, Jianhui Huang, and Jingping Bi. 2019. Cross-network embedding for multi-network alignment. In *The world wide web conference*. 273–284.

[4] Tyler Derr, Hamid Karimi, Xiaorui Liu, Jiejun Xu, and Jiliang Tang. 2019. Deep adversarial network alignment. *arXiv preprint arXiv:1902.10307* (2019).

[5] Chris Ding, Tao Li, and Michael I Jordan. 2008. Nonnegative matrix factorization for combinatorial optimization: Spectral clustering, graph matching, and clique finding. In *2008 Eighth IEEE International Conference on Data Mining*. IEEE.

[6] Jingtao Ding, Yuhan Quan, Quanming Yao, Yong Li, and Depeng Jin. 2020. Simplify and Robustify Negative Sampling for Implicit Collaborative Filtering. *arXiv preprint arXiv:2009.03376* (2020).

[7] Fazle E Faisal, Lei Meng, Joseph Crawford, and Tijana Milenković. 2015. The post-genomic era of biological network alignment. *EURASIP Journal on Bioinformatics and Systems Biology* 2015, 1 (2015), 1–19.

[8] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[9] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. 2018. Regal: Representation learning-based graph alignment. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 117–126.

[10] Bo Jiang, Jin Tang, Chris Ding, Yihong Gong, and Bin Luo. 2017. Graph matching via multiplicative update algorithm. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 3190–3198.

[11] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[12] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[13] Li Liu, William K Cheung, Xin Li, and Lejian Liao. 2016. Aligning Users across Social Networks Using Network Embedding.. In *Ijcai*. 1774–1780.

[14] M Maruf and Anuj Karpatne. 2021. Maximizing Cohesion and Separation in Graph Representation Learning: A Distance-aware Negative Sampling Approach. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM, 271–279.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546* (2013).

[16] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[17] Jiaxiang Ren, Yang Zhou, Ruoming Jin, Zijie Zhang, Dejing Dou, and Pengwei Wang. 2019. Dual adversarial learning based network alignment. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1288–1293.

[18] Steffen Rendle and Christoph Freudenthaler. 2014. Improving pairwise learning for item recommendation from implicit feedback. In *Proceedings of the 7th ACM international conference on Web search and data mining*. 273–282.

[19] Vikram Saraph and Tijana Milenković. 2014. MAGNA: maximizing accuracy in global network alignment. *Bioinformatics* 30, 20 (2014), 2931–2940.

[20] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*. Springer, 593–607.

[21] Rohit Singh, Jinbo Xu, and Bonnie Berger. 2007. Pairwise global alignment of protein interaction networks by matching neighborhood topology. In *Annual International Conference on Research in Computational Molecular Biology*. Springer.

[22] Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. 2018. Bootstrapping Entity Alignment with Knowledge Graph Embedding.. In *IJCAI*, Vol. 18.

[23] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 990–998.

[24] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast random walk with restart and its applications. In *Sixth international conference on data mining (ICDM'06)*. IEEE, 613–622.

[25] Vipin Vijayan, Vikram Saraph, and Tijana Milenković. 2015. MAGNA++: Maximizing Accuracy in Global Network Alignment via both node and edge conservation. *Bioinformatics* 31, 14 (2015), 2409–2411.

[26] Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. 2018. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

[27] Yuchen Yan, Si Zhang, and Hanghang Tong. 2021. BRIGHT: A Bridging Algorithm for Network Alignment. In *The World Wide Web Conference*.

[28] Zhen Yang, Ming Ding, Chang Zhou, Hongxia Yang, Jingren Zhou, and Jie Tang. 2020. Understanding negative sampling in graph representation learning. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1666–1676.

[29] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.

[30] Jiawei Zhang and S Yu Philip. 2015. Integrated anchor and social link predictions across social networks. In *Twenty-fourth international joint conference on artificial intelligence*.

[31] Jiawei Zhang and S Yu Philip. 2015. Multiple anonymized social networks alignment. In *2015 IEEE International Conference on Data Mining*. IEEE, 599–608.

[32] Si Zhang and Hanghang Tong. 2016. Final: Fast attributed network alignment. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1345–1354.

[33] Si Zhang and Hanghang Tong. 2018. Attributed network alignment: Problem definitions and fast solutions. *IEEE Transactions on Knowledge and Data Engineering* 31, 9 (2018), 1680–1692.

[34] Si Zhang, Hanghang Tong, Jie Tang, Jiejun Xu, and Wei Fan. 2020. Incomplete network alignment: Problem definitions and fast solutions. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 14, 4 (2020), 1–26.

[35] Si Zhang, Hanghang Tong, Yinglong Xia, Liang Xiong, and Jiejun Xu. 2020. NetTrans: Neural Cross-Network Transformation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[36] Dawei Zhou, Lecheng Zheng, Jiawei Han, and Jingrui He. 2020. A data-driven graph generative model for temporal interaction networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 401–411.

[37] Fan Zhou, Lei Liu, Kunpeng Zhang, Goce Trajcevski, Jin Wu, and Ting Zhong. 2018. Deeplink: A deep learning approach for user identity linkage. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 1313–1321.

[38] Yao Zhou, Jianpeng Xu, Jun Wu, Zeinab Taghavi, Evren Korpeoglu, Achan Kannan, and Jingrui He. 2021. PURE: Positive-Unlabeled Recommendation with Generative Adversarial Network. In *KDD*. ACM.

Reproducibility

Dataset descriptions. We evaluate the performance of network alignment in three different scenarios, and the datasets that we use to construct them are described as below.

- *S1 - ACM-DBLP [33].* In this scenario, we want to align two undirected co-author networks ACM and DBLP that are extracted from the papers in four areas (DM, ML, DB and IR) and their corresponding citation information [23]. In these co-author networks, nodes represent authors and there exists an edge between two nodes if they are co-authors of at least one paper. Specifically, the ACM co-author network has 9,872 nodes and 39,561 edges. The DBLP co-author network has 9,916 nodes and 44,808 edges. The attributes of each node indicate the number of papers that are published in different venues by that author. There exist 6,325 common authors across two networks used as the groundtruth alignments.
- *S2 - Foursquare-Twitter [30].* In this scenario, we want to align two social networks of Foursquare and Twitter. Each node represent a user and edges indicate the friendships among users. There are 5,313 nodes and 5,120 edges in the Foursquare network. And the Twitter network has 5,120 nodes and 130,575 edges. Node attributes are not available in these two networks. In addition, there are 1,609 common users which are used as the groundtruth alignments.
- *S3 - Phone-Email [34].* In this scenario, we aim to align the communication networks through different channels. In particular, the Phone network corresponds to the communications among people via phone, while the Email network describes the communications by emails. More specifically, there exist 1,000 nodes and 41,191 edges in the Phone network while the Email network is sparser with 1,003 nodes and 4,627 edges. In addition, there are 1,000 common people that are involved in both communication networks used as the groundtruth alignments.

Besides, in *S1-S3*, we evaluate with different training ratios (i.e., 10% and 20%). For example, with the training ratio as 10%, we randomly select 10% of the groundtruth alignments as the training data (i.e., anchor links) and test on the rest of the groundtruth alignments. We repeat and randomly generate 10 sets of training data for each alignment scenario. We evaluate the performance of all methods, and report the mean values and standard deviations.

Baseline methods. We compare the proposed method NeXtAlign with the following semi-supervised network alignment methods: (1) Bright [27], (2) NetTrans [35], (3) semi-supervised FINAL [32], (4) IONE [13], and (5) CrossMNA [3].

Machine and Repeatability. The proposed model is implemented in Pytorch. We use one Nvidia GTX 1080 as GPU. We will release the source code and the datasets after the paper is published.

Hyperparameters settings. We use Adam optimizer with a learning rate 0.05 to train the model. We use the same hyperparameter setting in all the three alignment scenarios. Specifically, we set $\alpha = 0.5, \lambda = 0.1$. In addition, we set the batch size as 300 and the number of negative samples as $k = 20$. We train the model in 50 epochs. For all embedding based methods, we learn node embeddings with the dimension $d = 128$. The parameters in all baseline methods are set to their defaults.

Metrics. We evaluate the effectiveness of network alignment in terms of Hits@K. Given a test pair (u, v), if node-v in $\mathcal{G}_2$ is among the top-K most similar nodes to node-u in $\mathcal{G}_1$, we view it as a *hit*. Then Hits@K is computed by $\text{Hits@}K = \frac{\text{\# of hits}}{\text{\# of testing alignments}}$.

Proof of Lemma 2

We prove Lemma 2 by extending the single network case [28].

Lemma. *Given an anchor link (a, x), the optimal embeddings that minimize $J_{(a,x)}$ satisfy for non-anchor nodes $b \in \bar{\mathcal{L}}_1, y \in \bar{\mathcal{L}}_2$,*

$$\mathbf{b}'\mathbf{x} = -\log \frac{kp_n(b|a) + kp_{nc}(b|x)}{p_d(b|a) + p_{dc}(b|x)}$$
$$\mathbf{y}'\mathbf{x} = -\log \frac{kp_n(y|x) + kp_{nc}(y|a)}{p_d(y|x) + p_{dc}(y|a)}$$

and for anchor nodes such that $(b, y) \in \mathcal{L}$,

$$\mathbf{b}'\mathbf{x} = \mathbf{y}'\mathbf{x} = -\log \frac{k[p_n(b|a) + p_{nc}(b|x) + p_n(y|x) + p_{nc}(y|a)]}{p_d(b|a) + p_{dc}(b|x) + p_d(y|x) + p_{nc}(y|a)}$$

Proof. For node-b in $\mathcal{G}_1$ and node-y in $\mathcal{G}_2$ such that $(b, y) \notin \mathcal{L}$, the main idea is to first prove that the loss functions Eq. (9) and Eq. (10) can be minimized separately by satisfying the conditions Eq. (11) and Eq. (12), and then prove these two conditions can co-occur. We first define two Bernoulli distributions $P_{b,(a,x)}(z = 1) = \frac{p_d(b|a)+p_{dc}(b|x)}{p_d(b|a)+p_{dc}(b|x)+kp_n(b|a)+kp_{nc}(b|x)}$ and $Q_{b,(a,x)}(z = 1) = \sigma(\mathbf{b}'\mathbf{a}) = \sigma(\mathbf{b}'\mathbf{x})$. Then the term of node-$b$ in Eq. (9) is

$$O_b = [p_d(b|a)+p_{dc}(b|x)+kp_n(b|a)+kp_{nc}(b|x)]H(P_{b,(a,x)}, Q_{b,(a,x)})$$

where $H(\cdot, \cdot)$ is the cross-entropy between two distributions. According to Gibbs Inequality, the minimum can be achieved when $P_{b,(a,x)} = Q_{b,(a,x)}$, $\forall b \in \mathcal{V}_1 - \{b|(b, y) \notin \mathcal{L}\}$. This implies

$$\mathbf{b}'\mathbf{x} = -\log \frac{kp_n(b|a) + kp_{nc}(b|x)}{p_d(b|a) + p_{dc}(b|x)}$$

Similarly, we can show the loss Eq. (10) is minimized when

$$\mathbf{y}'\mathbf{x} = -\log \frac{kp_n(y|x) + kp_{nc}(y|a)}{p_d(y|x) + p_{dc}(y|a)}$$

Since $(b, y) \notin \mathcal{L}$, it is easy to see that at least one of $\mathbf{b}$ and $\mathbf{y}$ could be an arbitrary vector as long as it satisfies the above condition.

Next, for two nodes such that $(b, y) \in \mathcal{L}$, since we share the embedding across the nodes of an anchor link (i.e., $\mathbf{b} = \mathbf{y}$), the corresponding term in Eq. (9) is equivalent to

$$O_b = -[p_d(b|a) + p_{dc}(b|x) + p_d(y|x) + p_{dc}(y|a)] \log \sigma(\mathbf{b}'\mathbf{x}) - [p_n(b|a) + p_{nc}(b|x) + p_n(y|x) + p_{nc}(y|a)] \log \sigma(-\mathbf{b}'\mathbf{x})$$

Similarly, we can derive the condition for $(b, y) \in \mathcal{L}$ as

$$\mathbf{b}'\mathbf{x} = \mathbf{y}'\mathbf{x} = -\log \frac{k[p_n(b|a) + p_{nc}(b|x) + p_n(y|x) + p_{nc}(b|a)]}{p_d(b|a) + p_{dc}(b|x) + p_d(y|x) + p_{nc}(y|a)}$$

□

Proof of Lemma 3

Next, we provide the proof for Lemma 3 as follows.

Lemma. *Denote $\Delta\boldsymbol{\theta}_b = \boldsymbol{\theta}_b^B - \boldsymbol{\theta}_b^*, \Delta\boldsymbol{\theta}_y = \boldsymbol{\theta}_y^B - \boldsymbol{\theta}_y^*$. The mean square errors for non-anchor nodes $b \in \bar{\mathcal{L}}_1$ and $y \in \bar{\mathcal{L}}_2$ is computed by*

$$\mathbb{E}\left[\Delta\theta_b^2\right] = \frac{1}{B}\left[\frac{1}{p_d(b|a) + p_{dc}(b|x)} + \frac{1}{kp_n(b|a) + kp_{nc}(b|x)} - C\right]$$
$$\mathbb{E}[\Delta\theta_y^2] = \frac{1}{B}\left[\frac{1}{p_d(y|x) + p_{dc}(y|a)} + \frac{1}{kp_n(y|x) + kp_{nc}(y|a)} - C\right]$$

For anchor nodes $b \in \mathcal{L}_1$ and $y \in \mathcal{L}_2$, the mean square error is

$$\mathbb{E}\left[\Delta\theta_b^2\right] = \mathbb{E}[\Delta\theta_y^2] = \frac{1}{B}\left[\frac{1}{p_1} + \frac{1}{kp_2} - C\right]$$

where $C = 1 + \frac{1}{k}$, $p_1 = p_d(b|a) + p_{dc}(b|x) + p_d(y|x) + p_{nc}(y|a)$ and $p_2 = p_n(b|a) + p_{nc}(b|x) + p_n(y|x) + p_{nc}(b|a)$.

Proof. By easing the notation $\nabla J^B_{(a,x)} = \nabla_{\boldsymbol{\theta}} J^B_{(a,x)}$ as the gradient, the optimal solution $\boldsymbol{\theta}^B$ implies $\nabla J^B_{(a,x)}(\boldsymbol{\theta}^B) = \mathbf{0}$, which gives

$$\nabla J^B_{(a,x)}(\boldsymbol{\theta}^B) = \nabla J^B_{(a,x)}(\boldsymbol{\theta}^*) + \nabla^2 J^B_{(a,x)}(\boldsymbol{\theta}^*)(\boldsymbol{\theta}^B - \boldsymbol{\theta}^*) + O(\|\boldsymbol{\theta}^B - \boldsymbol{\theta}^*\|^2) = \mathbf{0}.$$

Thus, up to terms of order $O(\|\boldsymbol{\theta}^B - \boldsymbol{\theta}^*\|^2)$, we have

$$\sqrt{B}(\boldsymbol{\theta}^B - \boldsymbol{\theta}^*) = -\left(\nabla^2 J^B_{(a,x)}(\boldsymbol{\theta}^*)\right)^{-1} \sqrt{B}\nabla J^B_{(a,x)}(\boldsymbol{\theta}^*)$$

Next we analyze $-\left(\nabla^2 J^B_{(a,x)}(\boldsymbol{\theta}^*)\right)^{-1}$ and $\sqrt{B}\nabla J^B_{(a,x)}(\boldsymbol{\theta}^*)$.

For $\left(\nabla^2 J^B_{(a,x)}(\boldsymbol{\theta}^*)\right)^{-1}$: The gradient and Hessian matrix of the empirical risk $J^B_{(a,x)}$ can be computed as

$$\begin{aligned}
\nabla J^B_{(a,x)}(\boldsymbol{\theta}) = &-\frac{1}{B}\sum_{i_1}(1-\sigma(\boldsymbol{\theta}_{b_{i_1}}))\mathbf{e}_{(b_{i_1})} - \frac{1}{B}\sum_{i_2}(1-\sigma(\boldsymbol{\theta}_{b_{i_2}}))\mathbf{e}_{(b_{i_2})} \\
&+ \frac{1}{B}\sum_{i_3}\sigma(\boldsymbol{\theta}_{b_{i_3}})\mathbf{e}_{(b_{i_3})} + \frac{1}{B}\sum_{i_4}\sigma(\boldsymbol{\theta}_{b_{i_4}})\mathbf{e}_{(b_{i_4})} \\
&- \frac{1}{B}\sum_{j_1}(1-\sigma(\boldsymbol{\theta}_{y_{j_1}}))\mathbf{e}_{(y_{j_1})} - \frac{1}{B}\sum_{j_2}(1-\sigma(\boldsymbol{\theta}_{y_{j_2}}))\mathbf{e}_{(y_{j_2})} \\
&+ \frac{1}{B}\sum_{j_3}\sigma(\boldsymbol{\theta}_{y_{i_3}})\mathbf{e}_{(y_{j_3})} + \frac{1}{B}\sum_{j_4}\sigma(\boldsymbol{\theta}_{y_{j_4}})\mathbf{e}_{(y_{j_4})} \\
\nabla^2 J^B_{(a,x)}(\boldsymbol{\theta}) = &\, f(b,i_1) + f(b,i_2) + f(b,i_3) + f(b,i_4) \\
&+ f(y,j_1) + f(y,j_2) + f(y,j_3) + f(y,j_4)
\end{aligned}$$

where for example $f(b,i_1) = \frac{1}{B}\sum_{i_1}\sigma(\boldsymbol{\theta}_{b_{i_1}})\left(1-\sigma(\boldsymbol{\theta}_{b_{i_1}})\right)\mathbf{e}_{(b_{i_1})}\mathbf{e}'_{(b_{i_1})}$ and $e_{(b_{i_1})}$ is a one-hot vector which has only a 1 on the corresponding dimension. According to Lemma 2, by denoting $\mathbf{H}_{(a,x)} = \lim_{B\to+\infty}\nabla^2 J^B_{(a,x)}(\boldsymbol{\theta}^*)$ we have at $\boldsymbol{\theta} = \boldsymbol{\theta}^*$

$$\begin{aligned}
\mathbf{H}_{(a,x)} \xrightarrow{P} &\sum_b \sigma(\boldsymbol{\theta}_b^*)(1-\sigma(\boldsymbol{\theta}_b^*))\mathbf{e}_{(b)}\mathbf{e}'_{(b)} \\
&\times [p_d(b|a) + p_{dc}(b|x) + p_n(b|a) + p_{nc}(b|x)] \\
&+ \sum_y \sigma(\boldsymbol{\theta}_y^*)(1-\sigma(\boldsymbol{\theta}_y^*))\mathbf{e}_{(y)}\mathbf{e}'_{(y)} \\
&\times [p_d(y|x) + p_{dc}(y|a) + p_n(y|x) + p_{nc}(y|a)] \\
= &\sum_{b\in\tilde{\mathcal{L}}_1} \frac{k[p_d(b|a)+p_{dc}(b|x)][p_n(b|a)+p_{nc}(b|x)]}{p_d(b|a)+p_{dc}(b|x)+p_n(b|a)+p_{nc}(b|x)}\mathbf{e}_{(b)}\mathbf{e}'_{(b)} \\
&+ \sum_{y\in\tilde{\mathcal{L}}_2} \frac{k[p_d(y|x)+p_{dc}(y|a)][p_n(y|x)+p_{nc}(y|a)]}{p_d(y|x)+p_{dc}(y|a)+p_n(y|x)+p_{nc}(y|a)}\mathbf{e}_{(y)}\mathbf{e}'_{(y)} \\
&+ \sum_{b\in\mathcal{L}_1} \frac{kp_1p_2}{p_1+p_2}\mathbf{e}_{(b)}\mathbf{e}'_{(b)} + \sum_{y\in\mathcal{L}_2} \frac{kp_1p_2}{p_1+p_2}\mathbf{e}_{(y)}\mathbf{e}'_{(y)} \\
= &\,\mathrm{diag}(\mathbf{m})
\end{aligned}$$

where $p_1 = p_d(b|a) + p_{dc}(b|x) + p_d(y|x) + p_{dc}(y|a)$ and $p_2 = p_n(b|a) + p_{nc}(b|x) + p_n(y|x) + p_{nc}(y|a)$.

We analyze $\nabla J^B_{(a,x)}(\boldsymbol{\theta}^*)$ expectation and variance as follows.

$$\begin{aligned}
\mathbb{E}[\nabla J^B_{(a,x)}(\boldsymbol{\theta}^*)] = &-\sum_{b\in\tilde{\mathcal{L}}_1}\Big[p_d(b|a)+p_{dc}(b|x)](1-\sigma(\boldsymbol{\theta}_b^*))\mathbf{e}_{(b)} \\
&- k[p_n(b|a)+p_{nc}(b|x)]\sigma(\boldsymbol{\theta}_b^*)\mathbf{e}_{(b)}\Big] \\
&- \sum_{y\in\tilde{\mathcal{L}}_2}\Big[p_d(y|x)+p_{dc}(y|a)](1-\sigma(\boldsymbol{\theta}_y^*))\mathbf{e}_{(y)} \\
&- k[p_n(y|x)+p_{nc}(y|a)]\sigma(\boldsymbol{\theta}_y^*)\mathbf{e}_{(y)}\Big] \\
&- \sum_{b\in\mathcal{L}_1} p_1(1-\sigma(\boldsymbol{\theta}_b^*))\mathbf{e}_{(b)} - kp_2\sigma(\boldsymbol{\theta}_b)\mathbf{e}_{(b)} \\
&- \sum_{y\in\mathcal{L}_2} p_1(1-\sigma(\boldsymbol{\theta}_y^*))\mathbf{e}_{(y)} - kp_2\sigma(\boldsymbol{\theta}_y)\mathbf{e}_{(y)} \\
= &\,\mathbf{0}
\end{aligned}$$

$$\begin{aligned}
\mathrm{Cov}[\nabla J^B_{(a,x)}(\boldsymbol{\theta}^*)] &= \mathbb{E}\left[\nabla J^B_{(a,x)}(\boldsymbol{\theta}^*)(\nabla J^B_{(a,x)}(\boldsymbol{\theta}^*))'\right] \\
&= \frac{1}{B}\left(\mathrm{diag}(\mathbf{m}) - (1+\frac{1}{k})\mathbf{m}\mathbf{m}'\right)
\end{aligned}$$

Then, with $\mathbf{H}_{(a,x)}$ and $\mathrm{Cov}[\nabla J^B_{(a,x)}(\boldsymbol{\theta}^*)]$, we can derive the covariance of $\sqrt{B}(\boldsymbol{\theta}^B - \boldsymbol{\theta}^*)$ as

$$\begin{aligned}
\mathrm{Cov}\left[\sqrt{B}(\boldsymbol{\theta}^B - \boldsymbol{\theta}^*)\right] &= \mathbb{E}\left[\sqrt{B}(\boldsymbol{\theta}^B - \boldsymbol{\theta}^*)\sqrt{B}(\boldsymbol{\theta}^B - \boldsymbol{\theta}^*)'\right] \\
&\approx B\mathrm{diag}(\mathbf{m})^{-1}\mathrm{Var}\left[\nabla J^B_{(a,x)}(\boldsymbol{\theta}^*)\right](\mathrm{diag}(\mathbf{m})^{-1})' \\
&= \mathrm{diag}(\mathbf{m})^{-1} - (1+\frac{1}{k})\mathbf{1}\mathbf{1}'
\end{aligned}$$

This implies that the mean square errors for non-anchor nodes $b \in \tilde{\mathcal{L}}_1$ and $y \in \tilde{\mathcal{L}}_2$ can be computed by

$$\mathbb{E}\left[\Delta\theta_b^2\right] = \frac{1}{B}\left[\frac{1}{p_d(b|a)+p_{dc}(b|x)} + \frac{1}{kp_n(b|a)+kp_{nc}(b|x)} - C\right]$$

$$\mathbb{E}[\Delta\theta_y^2] = \frac{1}{B}\left[\frac{1}{p_d(y|x)+p_{dc}(y|a)} + \frac{1}{kp_n(y|x)+kp_{nc}(y|a)} - C\right]$$

For anchor nodes $b \in \mathcal{L}_1$ and $y \in \mathcal{L}_2$, the mean square error is computed by

$$\mathbb{E}\left[\Delta\theta_b^2\right] = \mathbb{E}[\Delta\theta_y^2] = \frac{1}{B}\left[\frac{1}{p_1} + \frac{1}{kp_2} - C\right]$$

where $\Delta\boldsymbol{\theta}_b = \boldsymbol{\theta}_b^B - \boldsymbol{\theta}_b^*$, $\Delta\boldsymbol{\theta}_y = \boldsymbol{\theta}_y^B - \boldsymbol{\theta}_y^*$. This completes the proof. □