Structural representation learning for network alignment with self-supervised anchor links

Nguyen Thanh Toan[a], Pham Minh Tam[b], Nguyen Thanh Tam[c], Huynh Thanh Trung[d], Tong Van Vinh[b], Nguyen Quoc Viet Hung[d], Quan Thanh Tho[e,*]

[a]*Faculty of Information Technology, Ho Chi Minh City University of Technology (HUTECH), Ho Chi Minh City, Vietnam*
[b]*Hanoi University of Science and Technology, Vietnam*
[c]*Ecole Polytechnique Federale de Lausanne, Switzerland*
[d]*Griffith University, Australia*
[e]*Department of Software Engineering, Ho Chi Minh City University of Technology–Vietnam National University, Ho Chi Minh City 76000, Vietnam*

Abstract

Network alignment, the problem of identifying similar nodes across networks, is an emerging research topic due to its ubiquitous applications in many data domains such as social-network reconciliation and protein-network analysis. While traditional alignment methods struggle to scale to large graphs, the state-of-the-art representation-based methods often rely on pre-defined anchor links, which are unavailable or expensive to compute in many applications. In this paper, we propose NAWAL, a novel, end-to-end unsupervised embedding-based network alignment framework emphasizing on structural information. The model first embeds network nodes into a low-dimension space where the structural neighborhoodship on original network is captured by the distance on the space. As the space for the input networks are learnt independently, we further leverage a generative adversarial deep neural network to reconcile the spaces without relying on hand-crafted features or domain-specific supervision. The empirical results on three real-world datasets show that NAWAL significantly outper-

*Corresponding author
Email addresses: `nt.toan@hutech.edu.vn` (Nguyen Thanh Toan), `pminhtamnb@gmail.com` (Pham Minh Tam), `thanhtamhp@gmail.com` (Nguyen Thanh Tam), `h.thanhtrung@griffith.edu.au` (Huynh Thanh Trung), `vinhbachkhoait@gmail.com` (Tong Van Vinh), `quocviethung.nguyen@griffith.edu.au` (Nguyen Quoc Viet Hung), `qttho@hcmut.edu.vn` (Quan Thanh Tho)

forms state-of-the-art baselines, by over 13% of accuracy against unsupervised methods and on par or better than supervised methods. Our technique also demonstrate the robustness against adversarial conditions, such as structural noises and graph size imbalance.

Keywords: graph mining, graph matching, network alignment, network representation learning, network embedding

1. Introduction

Networks are universal languages for describing complex data. The network data structure naturally captures relationships between entities from different fields, such as social networks analysis, economics, bioinformatics and pattern recognition. Effective analyses on these data benefit a great range of subsequent research works and applications, including link prediction (Wang et al., 2018; Pandey et al., 2019), node classification (Gupta et al., 2017; Du et al., 2019) and community detection (Francisquini et al., 2017; Han et al., 2018; Yang et al., 2018b), which prove the importance of network data. However, these works only focus on modeling single networks whereas many real-world applications require interpreting data from multiple networks. Consequently, in recent years, network alignment, the task of identifying the correspondence of nodes across different networks, has been proposed as a prerequisite for many interesting inter-network applications (Bayati et al., 2009).

Network alignment is one of the fundamental problem in many intelligent data analysis applications. In urban planning and and smart city applications, some integrated expert systems are used to find the mappings of points of interest across different traffic network (Zhang & Philip, 2015). In robotic intelligence systems, network alignment helps to match difference scenes without human supervision (Yang et al., 2018a). In social networks, we expect the aligned nodes to be held by the same person. This predictive information can be exploited to perform better downstream functions such as social recommendation (Nisha & Mohan, 2019), content recommendation (Zhang & Tong, 2016), and group recommendation (Toan et al., 2018). In bioinfomatics, knowledge-based expert systems that identify the common protein molecules between different protein-protein-interaction (PPI) networks (Hashemifar & Xu, 2014) will be indispensable for the new integrative systems biology.

Despite its huge benefits to expert and intelligence systems, with the nature of an NP-hard problem (Bayati et al., 2009), network alignment is a challenging task. Many unsupervised approaches have been proposed to solve this problem directly, such as IsoRank (Singh et al., 2008), NetAlign (Bayati et al., 2009), UniAlign (Koutra et al., 2013), FINAL (Zhang & Tong, 2016) and REGAL (Heimann et al., 2018). Although having a straightforward advantage as not using any labelled pairs, these methods often rely on matrix decomposition, whose assumptions are not always applicable for domain-specific networks with unique structures such as social networks, where neighborhood relations (e.g. friendship) matter. Moreover, the fail to deal with large-scale networks, as the matrix decomposition on the whole network is often polynomial (cubic) (Bayati et al., 2009).

To make the solution scalable, new supervised alignment techniques (Man et al., 2016) leverage existing network embeddings (Perozzi & Skiena, 2014; Grover & Leskovec, 2016; Hamilton et al., 2017; Pham & Do, 2019) to compute the alignment function directly from the latent node features. However, they often rely on a large amount of labelled data for training the latent features, which requires heavy manual labor and nevertheless domain-specific only (Zhou et al., 2018). Some methods try to mitigate the issue by using only pre-defined anchor links (i.e. important pairs of nodes known prior by domain knowledge) for training, which is not always available and might still fail to capture other links if network structures are not uniform (Liu et al., 2019). For example, in social networks, users have multiple accounts in different online platforms but often have little or no motivation to explicitly correlate their identities (Cao et al., 2018). To sum up, current methods have either not reached competitive performance to handle large-scale network or they still require a large amount of explicit anchor links to deal with network alignment problems efficiently.

In this paper, we introduce a scalable and autonomous framework entitled **N**etwork **A**lignment **W**ith Self-supervised **A**nchor **L**inks (**NAWAL**) that go beyond unsupervised and supervised approaches. NAWAL is *scalable* by leveraging the advantage of learning the light-weight node features from neighborhood/proximity structures of the network itself to overcome the efficiency issue of unsupervised methods. However, unlike supervised techniques, NAWAL defines its own anchor links from the current top similar pairs and refines the matching from these anchor links. This process is repeated many times to reach the optimum (e.g. the new anchors is not different from the previous ones).

More precisely, we first embed independently the source and target networks into low-dimensional latent spaces. The advantages of this embedding step are two-fold: (1) it helps to the facilitate the computation for later process thanks to low-dimension characteristic of the latent spaces and (2) the learned embeddings capture the topology trait of each network node by using distance (similarity in the embedding space approximates similarity in the original network), which can be used to distinguish one node from the others. Next, as the embedding spaces are constructed independently, a mapping function $\mathbf{W}$ is learned to reconcile the two generated embedding spaces into a common space where the corresponding node pairs from source and target networks have similar embedding. To achieve a high quality mapping function without supervision data, first we initialize the mapping by employing a generative adversarial network (Goodfellow, 2016), then carefully further refine it by applying the closed-form Procrustes solution (Schönemann, 1966) on high confidence anchor links. Finally, with the optimized learned mapping $\mathbf{W}^*$, we leverage a greedy heuristic to seek for all hidden node pairs between networks.

The main contributions are summarized as follows:

- We propose an embedding-based **N**etwork **A**lignment **W**ith Self-supervised **A**nchor **L**inks (**NAWAL**), an unsupervised network alignment framework which identifies node alignments by first learning node embeddings for each network, then obtaining hidden node pairs by reconciling the embedding spaces using a sophisticated mapping function. At the level of 10% network structural noise, NAWAL reaches over 90% whereas other unsupervised approaches achieve just less than 50% on average.

- Within our framework, to achieve high-quality mapping without using any cross-network supervision information, we first apply the GAN network to initialize the mapping which is able to distinguish between embeddings of source and target spaces, then fine-tune the mapping with the closed-form Procrustes solution. To the best of our knowledge, we are the first to do so to tackle the network alignment problem.

- Experiments on real-world graphs show that our approach success to achieve solid alignment results on three different real-world datasets and exhibits robustness to noise factors such as structural inconsistency and graph size imbalance. Our technique significantly outperforms all unsupervised baselines with over 13% accuracy higher than the second best technique (FINAL) and achieves similar performance against supervised baselines without using any prior knowledge (labelled data or pre-defined anchor links), especially reaching the same or better level of performance of the best supervised method (PALE) in many scenarios.

The remaining sections of our paper are organized into nine sections. Section 2 introduces some necessary background information on our study. Section 3 provides an overview of our approach. Section 4 outlines how to embed single networks independently, whereas Section 5 shows how to reconcile embedding spaces in the network alignment stage. Section 6 reveals the alignment result retrieval technique, and then experimental results are presented in Section 7. Section 8 discusses related work before we conclude in Section 9.

2. Background

In this section, as background, we define some important terminologies that are used to discuss throughout this paper.

Definition 1: ***Network.*** *A network is a pair* $\mathbf{G} = (\mathbf{V}, \mathbf{E})$*, where* $\mathbf{V}$ *is a set of nodes together with* $\mathbf{E}$ *is a set of edges.*

This definition can be used to generalize most networks by changing the detailed definition of $\mathbf{V}$ and $\mathbf{E}$. For example, in an online social network, $\mathbf{E}$ contains social relationships or tracking relationships and $\mathbf{V}$ represents users in the network. We learn a unique latent user space learning for each network according to a network embedding technique. Based on this, we define network embedding as follow:

Definition 2: ***Network Embedding.*** *Given a network* $\mathbf{G} = (\mathbf{V}, \mathbf{E})$*, network embedding aims to learn an encoder function* $\mathbf{ENC}_\Theta : v \rightarrow z_v \in \mathbf{R}^d$*, where* z_v *is the latent representation of vertex* v *and* d *is the dimension of the learned representation. The encoder* $\mathbf{ENC}_\Theta$ *retains the original network information, such that two vertices similar in the original network should have similar representations in the learned vector space. Learning node embeddings requires three main steps:*

- *Define an encoder* $\mathbf{ENC}_\Theta : \mathbf{V} \rightarrow \mathbf{R}^d$ *projects each network node to a low-dimensional space.*
- *Define a similarity function* $\mathbf{S}_G : \mathbf{V} \times \mathbf{V} \rightarrow \mathbf{R}^+$ *in the original network that measures the relationship between any two nodes.*
- *Optimize the parameters* Θ *of the encoder so that similarity function* $\mathbf{S}_G$ *of two nodes approximates the distance between their representation vectors:*

$$\mathbf{S}_G(u, v) \approx \mathbf{ENC}_\Theta(u)^T \mathbf{ENC}_\Theta(v); u, v \in \mathbf{V} \tag{1}$$

These embeddings then can be used as features for a wide range of machine learning tasks such as classification, clustering, link prediction, and visualization.

Definition 3: ***Anchor Links.*** *Given two networks* $\mathbf{G}_s = (\mathbf{V}_s, \mathbf{E}_s), \mathbf{G}_t = (\mathbf{V}_t, \mathbf{E}_t)$ *are aligned networks. We say the node pair* (a_s, a_t) *where* $a_s \in \mathbf{V}_s, a_t \in \mathbf{V}_t$ *is an anchor link, if* $a_s = a_t$*.*

To distinguish the nodes in the source and target networks, we set s and t as the lower notation for each network. The number of anchors has a great influence on the ability to reconcile the networks. The larger the number of anchors, the higher chances of reconciling them.

3. Model and Approach

In this section, we propose the NAWAL Framework for the network alignment problem in general. We first introduce a motivating example in Section 3.1, then define the network alignment in Section 3.2. The overview of our approach is described in Section 3.3.

3.1. Motivating Example

In this section, we introduce a motivating example that serves as the running example of this paper. Fig.1 considers a setting in which a system desires to predict whether a pair of users belong to the same real person. In other words, we need to identify all virtual twins between social networks, such as Facebook and Twitter. It is worth noting that little to no prior knowledge about a pair of users is given. Likewise in the figure, only one aligned pair is known in advance whereas many other pairs are still hidden.

In the above example, although some users may expose their explicit information in a third-party channel (like www.aboutme.com site), this information is limited. It would be helpful if all hidden pairs of users are extracted which supports a variety of applications. In addition, the approach is more adaptive to the real-world scenario if all anchor links are predictable although limited resources are available. Such predictive problem can be formulated as a network alignment problem where its formal definition is presented in the next section. Solving the network alignment under a very low resource constraint as in this example motivates us to propose the idea of this paper.

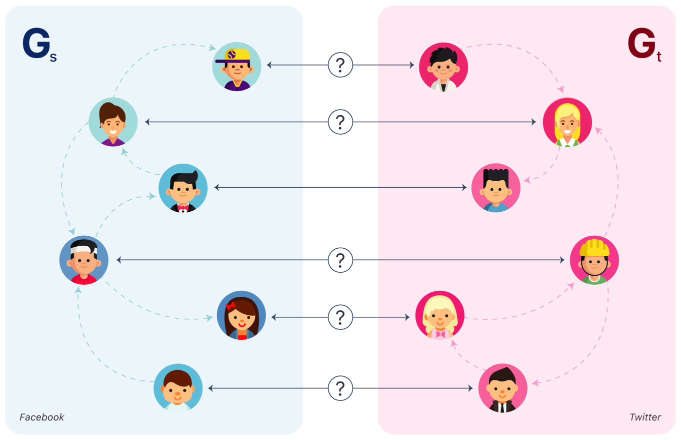


Figure 1: A motivating example of network alignment

3.2. Model

In this section, we formally define the problem of network alignment in general. We focus on one-to-one alignment problem, where each node can only be aligned to a maximum of one corresponding node in parallel networks (Kong et al., 2013). For simplicity, we focus on aligning two graphs (e.g., social or protein networks), although our method can easily be extended to more networks. Without loss of generality, we select one network as source network and the other as target network, denoted by $\mathbf{G}_s$ and $\mathbf{G}_t$ respectively. For each node in the source network, we aim to recognize, if any, its counterpart in the target network. To achieve this goal, network alignment techniques often calculate an alignment matrix $\mathbf{S}$, which is technically a cross-network similarity matrix: $\mathbf{S}(u, v)$ represents the similarity between a node $u \in \mathbf{V}_s$ and $v \in \mathbf{V}_t$. This can be formally formulated as follow:

Network alignment*. Given two networks $\mathbf{G}_s = (\mathbf{V}_s, \mathbf{E}_s), \mathbf{G}_t = (\mathbf{V}_t, \mathbf{E}_t)$ where $\mathbf{V}_s, \mathbf{V}_t$ are sets of nodes and $\mathbf{E}_s, \mathbf{E}_t$ are sets of edges, the problem of network alignment is to return an alignment matrix $\mathbf{S}$ where $\mathbf{S}(u, v)$ represents the similarity between a node $u \in \mathbf{V}_s$ and $v \in \mathbf{V}_t$.*

Matching node pairs across source and target network then can be inferred by applying heuristics on this alignment matrix (Heimann et al., 2018; Kollias et al., 2012) to learn $\mathbf{M} : \mathbf{V}_s \times \mathbf{V}_t \rightarrow \{0, 1\}$ such that $\mathbf{M}(u, v) = 1$ if two nodes $u \in \mathbf{V}_s$, $v \in \mathbf{V}_t$ share the same identity; otherwise $\mathbf{M}(u, v) = 0$.

The summary of notations used throughout the paper and their usages can be found in Table 1.

Table 1: Summary of notations

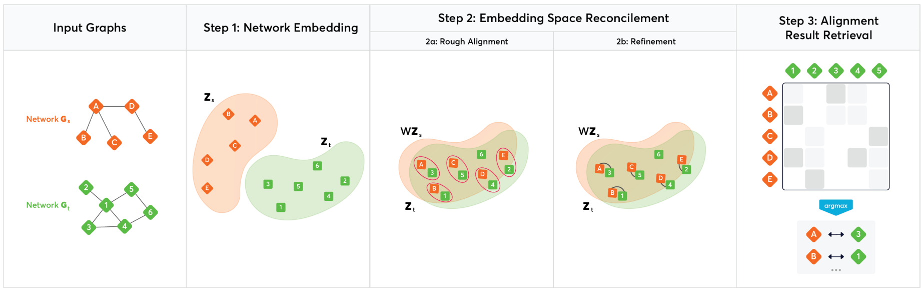
Notation	Description	Usages
$\mathbf{W}$, $\mathbf{W}_{init}$	The alignment matrix that maps the source network's embedding to the same vectoral space to the target network's embedding.	Section 5.1, 5.2
$\mathbf{W}^*$	The optimal alignment matrix after being refined by applying the closed-form Procrustes solution.	Section 5.2,6
$\mathbf{M}(u, v)$	The matrix (with only 0 or 1 value) represents whether u and v are aligned nodes.	Section 3.2,6
$\mathbf{S}(u, v)$	The matrix represents the cross-network similarity of two nodes between graphs.	Section 3.2,6

3.3. Approach

In literature, the traditional network alignment techniques using matrix factorization to directly calculate the alignment matrix can work with no supervision data whereas achieve good accuracy for some cases. However, they fail to scale to large network due to the sparsity and the enormous size of adjacency matrix. Some recent works leverage the advances of network embedding by projecting the network nodes to a low-dimension vector space to make their model adapt to such large network. As the representation spaces are learned independently, these models require a number of observed links between nodes of the two networks as anchors to reconcile the embedding spaces so that after reconciling to a common space, the embeddings of the anchor nodes will be close. However, these kind of supervised data are unavailable or hard to obtain in many applications because of the privacy policy of the service providers or the unwilling of users of revealing their identity in another platform. Therefore, in our work, we propose a network alignment model without using any supervision information.

There are many challenges we have to consider to overcome this problem. First, the model has to guarantee scalability, which is essential for modern applications. As a result, we choose to apply network embedding as the immediate step to reduce the dimension of the representation spaces and thus make the model scalable. The low-dimension embedding spaces also capture the characteristic of network nodes (i.e., homophily principle), which helps to identify the nodes across source and target networks. However, choosing representation learning poses the second challenge: the incompatibility between the representation space of the source and the target network. Although the source and target networks are encoded by the same embedding technique, these embeddings might belong to different and incomparable vector spaces as the two encoding processes are independent. Therefore, after obtaining the node embeddings for each network, we learn a mapping function to reconcile the two embedding spaces into one common space such that the corresponding nodes in two networks will have similar embeddings in this common space. However, learning the mapping function without any supervision data itself is a hard challenge that has never be addressed before in the network alignment literature. Given this challenge, there is a need of a special mechanism that can interpolate the mapping function from the relative similarity of the structure between two representation spaces.

In the light of the above considerations, we design our unsupervised representation based network alignment framework to contain 3 components as shown in Fig.2. The aim of this study is to develop an intelligent framework using deep learning to enrich the knowledge representation by aligning network representation from multiple domains and the whole process is autonomous (i.e., without any help or supervision from human or knowledge expert).



Input Graphs
Step 1: Network Embedding
Step 2: Embedding Space Reconcilement
2a: Rough Alignment
2b: Refinement
Step 3: Alignment Result Retrieval
Network G_s
Network G_t
Z_s
Z_t
WZ_s
argmax

Figure 2: Overview of NAWAL Framework.

3.3.1. *Network embedding*

In this step, we embed the source and target network independently into low-dimension latent spaces. To do that, we maximize the co-occurrence of direct neighbor nodes using log-likelihood function and negative sampling, the popular strategy which helps to speed up the training process. We consider the first-order topology information instead of the higher-order (e.g unbiased random walks, biased random walks) for two reasons: (1) The inconsistency due to random sampling often amplifies the difference between the representation spaces of the networks, which consequently makes them harder, even impossible to reconcile; and (2) they can raise computational and time expense due to the sampling step, especially for biased random walks. More detail of this step is described in Section 4.

3.3.2. *Reconciling embedding spaces*

This main focus of this component is to reconcile the learned embedding spaces by learning a mapping function without using any supervision data. The learning function we choose here is the linear function, as this is simple but effective model achieved good results on the word translation task, on par with a more complicated model such as multi-layer perceptron (Mikolov et al., 2013a). To train the linear mapping in an unsupervised way, we first generate the initial mapping using the generative adversarial network. The network simulates a two-player game: the generator tries to generate the mapping function that reconciles the networks whereas the discriminator tries to distinguish them. Then, we use the nodes that align the best as the initial anchor points to gradually train the mapping function. Further detail of this step will be described in Section 5.

3.3.3. Retrieving alignment result

After learning the mapping function to reconcile the mapping spaces, we obtain the alignment results in this stage. Intuitively, the nodes which have similar embeddings will be matched together. In consideration of this belief, we leverage a greedy heuristic to align all nodes between graphs. To facilitate the searching for the nearest neighbor algorithms, an efficient data structure has been proposed to fast identify the topmost similar embeddings among graphs. Further detail of this step is described in Section 6.

4. Network Embedding

In this section, we embed separately the given networks into low-dimensional spaces which encapsulates the intrinsic structure of the graph. By doing so, the latent representation of network maintains the following criteria:

- *Dimensional reduction.* The dimension of the representation space is much smaller than the number of network edges. This helps to facilitate many subsequent network analysis task both in term of computational efficiency and scalability, as the adjacency matrix of the large real-world network is often sparse and massive size.
- *Informative.* The node embeddings reflect node proximity of the network nodes in original network in terms of topology structure. This means that any two nodes have direct or neighborhood relationship should have similar embeddings.
- *Continuous.* The learned embeddings have continuous values to support subsequent analytic where these embeddings are used as features. The embeddings then become the self-identity of network nodes, which helps to identify the corresponding nodes across the networks.

As the networks are embedded independently, for simplicity, we focus on describing the embedding on one network $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ without distinguishing between the source and the target networks. For every pair of nodes $(u, v) \in \mathbf{E}$, the goal is to find the parameters θ so as to maximize their co-occurrence probability between them:

$$\underset{\theta}{\text{maximize}} \prod_{(u,v)\in\mathbf{E}} p(u|v;\theta) \tag{2}$$

For network embedding, the parameters θ are their vector representations (z_u, z_v). Let us denote by $p(E = 1|u, v)$ the probability that (u, v) came from the edge sets. Our objective function to best reflect the co-occurrence of nodes in edges set will become:

$$\underset{\theta}{\text{maximize}} \prod_{(u,v)\in\mathbf{E}} p(E = 1|u, v;\theta) \tag{3}$$

To reduce the computation complexity, the log function is applied to transform the product to sum. The objective function then becomes maximizing the log-likelihood:

$$\underset{\theta}{\text{maximize}} \sum_{(u,v)\in\mathbf{E}} \log p(E = 1|u, v;\theta) \tag{4}$$

where $p(E = 1|u, v; \theta)$ can be defined using softmax as in the skip-gram model (Mikolov et al., 2013b):

$$p(E = 1|u, v;\theta) = \frac{1}{1 + e^{-z_u^T.z_v}} = \sigma(z_u^T.z_v) \tag{5}$$

leading to the objective:

$$\underset{\theta}{\text{maximize}} \sum_{(u,v)\in\mathbf{E}} \log \sigma(z_u^T.z_v) \tag{6}$$

Since only observed edges are taken into account in the above objective function, there exists a trivial solution when $z_u = z_v$ and $z_u^T.z_v \longrightarrow \infty$. To avoid this phenomenon, for each edge (u, v), the nodes $v_k|(u, v_k) \notin \mathbf{E}$ are chosen and the probability $p(E = 0|u, v_k)$ is added to the objective function, with

$p(E=0|u,v_k) = 1 - p(E=1|u,v)$ being the probability that (u, v_k) do not come from the edge sets:

$$\underset{\theta}{\text{maximize}} \;\; \log[\sigma(z_u^T.z_v)] + \sum_{k=1}^{K} \log[(1 - \sigma(z_u^T.z_k))] \tag{7}$$

where there are K negative samples. Empirically, each node is sampled by the probability $\mathbf{P}(v) \sim d_v^{3/4}$ as proposed in (Mikolov et al., 2013a) with d_v is the degree of node v. Finally, we train the representation vector independently for all nodes of network $\mathbf{G}$ with a stochastic gradient descent algorithm.

5. Reconciling Embedding Spaces

In this section, we consider that we have two embedded sets ($\mathbf{Z}_s \leftarrow \mathbf{G}_s, \mathbf{Z}_t \leftarrow \mathbf{G}_t$) trained independently according to the method proposed in Section 4. Let $\mathbf{Z}_s = \{z_s^1, ..., z_s^n\}$ and $\mathbf{Z}_t = \{z_t^1, ..., z_t^m\}$ be two sets of n and m node embeddings which come from the source and target networks respectively. We focus on finding a mapping $\mathbf{W}$ between two sets so that *translations* are close to each other in the shared space. We use the term *translation* here as this technique is motivated by (Conneau et al., 2017) as an idea of machine translation. We propose a GAN-based approach to learn this mapping $\mathbf{W}$ between the source and target networks. We accomplish this by using both generative and discriminative processes as presented in Fig.3.

5.1. Rough alignment under a GAN-based setting

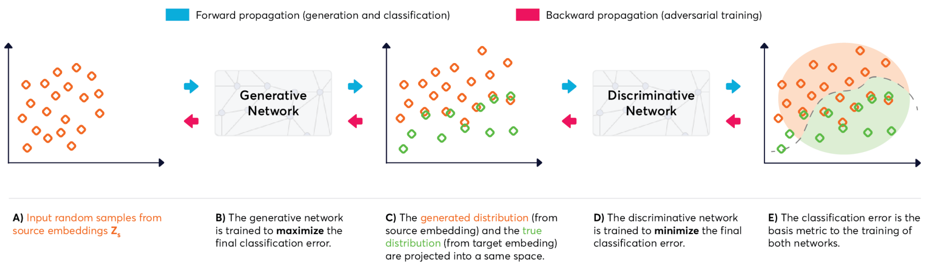

Forward propagation (generation and classification)
Backward propagation (adversarial training)
Generative Network
Discriminative Network
A) Input random samples from source embeddings Z_s
B) The generative network is trained to **maximize** the final classification error.
C) The generated distribution (from source embedding) and the true distribution (from target embeding) are projected into a same space.
D) The discriminative network is trained to **minimize** the final classification error.
E) The classification error is the basis metric to the training of both networks.

Figure 3: A GAN-based approach to learn the mapping $\mathbf{W}$ between the source and target networks.

The basic idea of a generative adversarial network (GAN) is a game between two players: the generator and the discriminator. Given a data distribution which we are trying to model p_{data}, the main goal of the first player, the generator, is to generate samples in p_{model} which approximates p_{data}. At the same time, the other player, the discriminator, examines samples to determine whether they are real or fake data. The discriminator is trained as a traditional binary classifier under a supervised setting by dividing inputs into two classes: *real class* ($l = 0$) or *fake class* (l = 1).

We employ the idea of GAN to make it suitable for our alignment problem. Formally, we design our framework as a structured probabilistic model (Goodfellow et al., 2016) that contains latent variables z_s sampled from $\mathbf{Z}_s$ and observed variables z_t sampled from $\mathbf{Z}_t$. This approach is in line with an unsupervised domain adaption problem as proposed by (Ganin et al., 2016), wherein our case, a domain is represented by a network embedding (source or target). The two players in the game are exposed by two processes which are differentiable by their inputs and parameters:

- The discriminator is presented as a function that takes samples (both real and fake samples) as inputs and $\mathbf{\Theta}_D$ as parameters.

- The generator is a function that takes z_s as an input and uses $\mathbf{W}$ as a parameter. It is worth noting that the learned $\mathbf{W}$ of the second player will be used as the mapping function for network alignment. Hence, from now on, we call this process an aligner instead of a generator as described in the original GAN concept.

Both processes have cost functions defined on $\mathbf{\Theta}_D$ and $\mathbf{W}$:

- The cost function for the discriminator is $\mathbf{J_D}(\mathbf{\Theta}_D, \mathbf{W})$. By minimizing the $\mathbf{J}_D$, we train $\mathbf{\Theta}_D$ to maximize the probability of assigning correct labels to both training examples and samples from the generator. This process controls only $\mathbf{\Theta}_D$ as parameters and is independent on $\mathbf{W}$. The discriminator loss function can be written as:

$$\mathbf{J}_D(\mathbf{\Theta}_D|\mathbf{W}) = -\frac{1}{n}\sum_{i=1}^{n}\log_{\mathbf{P}_{\mathbf{\Theta_D}}}(\mathrm{l}=1|\mathbf{W}z_s^i) - \frac{1}{m}\sum_{j=1}^{m}\log_{\mathbf{P}_{\mathbf{\Theta_D}}}(\mathrm{l}=0|z_t^j) \quad (8)$$

where $z_s^i \in \mathbf{Z}_s, z_t^j \in \mathbf{Z}_t$.

- The cost function for the aligner is $\mathbf{J}_W(\mathbf{\Theta}_D, \mathbf{W})$. This process controls only $\mathbf{W}$ as parameters and is independent on $\mathbf{\Theta}_D$. In the unsupervised setting, W is trained so that the discriminator is unable to distinguish between generated and actual data. The aligner loss function can be written as:

$$\mathbf{J}_W(\mathbf{W}|\mathbf{\Theta}_D) = -\frac{1}{n}\sum_{i=1}^{n}\log_{\mathbf{P}_{\mathbf{\Theta_D}}}(\mathrm{l}=0|\mathbf{W}z_s^i) - \frac{1}{m}\sum_{j=1}^{m}\log_{\mathbf{P}_{\mathbf{\Theta_D}}}(\mathrm{l}=1|z_t^j) \quad (9)$$

where $z_s^i \in \mathbf{Z}_s, z_t^j \in \mathbf{Z}_t$.

Orthogonality. The aligner $\mathbf{W}$ obtained in the previous step gives a good performance. However, (Smith et al., 2017) showed that we can further improve the result by enforcing $\mathbf{W}$ orthogonal. They showed that the optimal linear transformation between vector spaces should be orthogonal as orthogonal matrices
help maintain the distance between any two vectors across spaces. In our case, we also want to preserve the dot product of embeddings which forms a more stable training algorithm.

Although SVD guarantees us to achieve an optimal result by enforcing the constraint of orthogonality, this is an expensive procedure when we compute iteratively together with stochastic gradient descent. Furthermore, it may guide the parameters to go far from the optimizing direction of the main gradient descent. Hence, we use an approximate approach to make the algorithm more efficient (Cisse et al., 2017). In particular, after every main update, we perform the following update:

$$\mathbf{W} \leftarrow (1+\beta)\mathbf{W} - \beta(\mathbf{W}\mathbf{W}^{\mathbf{T}})\mathbf{W} \quad (10)$$

Empirically, we found that beta = 0.01 works best with our framework.

Algorithm 1 Rough alignment algorithm

```
1: Input: Embeddings of node in source network $\mathbf{Z}_s$
2:        Embeddings of node in target network $\mathbf{Z}_t$
3:        Number of training iterations $T$
4:        Number of step per discriminator train $K_{dis}$
5:        Batch size $m$
6: Output: Initial linear mapping $\mathbf{W}_{init}$
7: $\mathbf{W} = INIT\_WEIGHT()$                        ▷ Initialize the mapping
8: for $t = 1 \ldots T$ do
9:     for $k = 1 \ldots K_{dis}$ do                    ▷ Training discriminator
10:        Sample $B_s^{dis} = \{z_s^1, ..., z_s^m\}$ from $\mathbf{Z}_s$.
11:        Sample $B_t^{dis} = \{z_t^1, ..., z_t^m\}$ from $\mathbf{Z}_t$.
12:        Update $\mathbf{\Theta}_D$ with $B_s^{dis}, B_t^{dis}$ by Eqn. 8.
13:    Sample noise samples $B^{gen} = \{z_s^1, ..., z_s^m\}$ from $\mathbf{Z}_s$.
14:    Update $\mathbf{W}$ with $B^{gen}$ by Eqn. 9.          ▷ Updating generator
15:    Enforce $\mathbf{W}$ by orthogonal constraint with Eqn. 10.
16: $\mathbf{W}_{init} \leftarrow \mathbf{W}$   ▷ Take the final result as initial mapping for refinement step
17: return $\mathbf{W}_{init}$
```

The training process. We follow the standard training procedure of (Goodfellow et al., 2014) to train our model (Algorithm 1). The training process consists of a simultaneous mini-batch gradient descent. On each step, two mini-batches are sampled: a mini-batch of values from the target embedding and a mini-batch of values drawn from the source embedding. Then two gradient steps are made simultaneously: one updating $\mathbf{\Theta}_D$ to reduce $\mathbf{J}_D$ and one updating $\mathbf{W}$ to reduce $\mathbf{J}_W$. For each iteration, $\mathbf{W}$ is also updated according to orthogonal constraints after the main gradient update.

After this section (Section 5.1), we get an initial mapping $\mathbf{W}$ which is able to roughly reconcile the two embedding spaces. This mapping will be further refined so that we denote it as $\mathbf{W}_{init}$ and it will used as the input for the next Section (Section 5.2).

5.2. Refinement

After the process introduced in the previous section, we obtain an initial linear mapping $\mathbf{W}_{init}$ that is able to align the vector representations of the source and target networks to some extent. However, the quality of this mapping function can be further improved due to two reasons. First, the generated function $\mathbf{W}_{init}$ is just approximately orthogonal, while this is a necessary characteristic for a good mapping (Cisse et al., 2017). Second, the GAN network attempts to align every node from source to target network, while there are some nodes with less topology information (i.e., low degree) can adversely affect the quality of the mapping. Therefore, in this section, we fine-tune the obtained matrix $\mathbf{W}_{init}$ from the previous step to achieve the finest mapping function.

To this end, starting from the initial mapping function, we select the most reliable matching pairs as the anchor links. To do so, we first employ cross-domain similarity local scaling (CSLS) as the comparison metric to produce matching pairs, as this metric has been proven by (Conneau et al., 2017) to be better in inducing pairs of the nearest neighbors than employing traditional Euclidean distance as the comparison metric. Then, only aligned pairs with a high matching score are taken as anchors to ensure the quality. We denote the selected anchor links set by $\mathbf{C}$.

After carefully choosing the anchor links set $\mathbf{C}$, we use this set as a groundtruth to train a new linear mapping function by minimizing the difference between the embedding of source nodes after applying the mapping function and the embedding of target nodes in the anchor links. Mathematically, we find the linear mapping matrix $\mathbf{W}^*$ such that:

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \; ||\mathbf{W}_{init}\mathbf{Z}_s - \mathbf{Z}_t||_F \tag{11}$$

where $(\mathbf{Z}_s, \mathbf{Z}_t)$ are embeddings of the nodes in anchor links set $\mathbf{C}$. Using the result from (Schönemann, 1966), the exact solution $\mathbf{W}^*$ in Equation 11 can be found using the following formula:

$$\mathbf{W}^* = \mathbf{U}\mathbf{V}^T \tag{12}$$

where $\mathbf{U}\Sigma\mathbf{V}^T = \mathbf{SVD}(\mathbf{Z}_t\mathbf{Z}_s^T)$. It is worth noting the singular value decomposition is performed on the anchor links set $\mathbf{C}$ only (whose size is extremely less than the data size) and thus tractable in practice. The algorithm for the refinement step is demonstrated in Algorithm 2.

Algorithm 2 Refinement algorithm

```
 1: Input: Embeddings of node in source network $\mathbf{Z}_s$
 2:        Embeddings of node in target network $\mathbf{Z}_t$
 3:        Initial linear mapping $\mathbf{W}_{init}$
 4:        Number of refinement step $K_{refine}$
 5:        Keeping ratio $k\_ratio$
 6: Output: Fine-tuned linear mapping $\mathbf{W}^*$
 7: $\mathbf{W}^* = \mathbf{W}_{init}$                    ▷ Initialize the optimal mapping
 8: for $t = 1 \dots K_{refine}$ do
 9:     $\mathbf{C}_{all}$ = Produce all matching pairs from $\mathbf{W}^*\mathbf{Z}_s$ and $\mathbf{Z}_t$ based on CSLS
10:     $\mathbf{C}$ = Select $k\_ratio$ of top matching anchor pairs from $\mathbf{C}_{all}$
11:     $\mathbf{W}^* = PROCRUSTES(\mathbf{C})$       ▷ Find optimized mapping with $\mathbf{C}$ using
    Eqn. 12.
12: return $\mathbf{W}^*$
```

6. Alignment Result Retrieval

The aligner $\mathbf{W}^*$ trained in the previous section is used to infer the similarity matrix as:

$$\mathbf{S} = (\mathbf{W}^*\mathbf{Z}_s)\mathbf{Z}_t^T \tag{13}$$

After getting the similarity matrix $\mathbf{S}$, a heuristic greedy matching algorithm (Kollias et al., 2012) is applied to the matrix as a post-processing step to achieve one-to-one alignment accuracy. The heuristic iteratively finds the highest score $\mathbf{S}[i, j]$ on the similarity matrix and records the node pair $\mathbf{M}(i, j) = 1$, then all scores involving either node i or node j are deleted from the matrix (and replaced with a zero value); the process stops when one of the graphs has all of its nodes paired.

In order to compute all pairs of similarities between the embeddings, a naive approach can be applied. To align one node pair, that approach needs to iterate all row pairs between embedding matrices to choose the top-1 similarity node. This is undesirable to do because of its computation complexity. In practice, only some of the top most-likely alignments should be considered. Hence, we need a better way for quickly finding the closest vectors which yields some specialized data structures that facilitate searching for the nearest neighbor algorithms and many other applications (Bhatia et al., 2010). Among those specialized data structures, we choose the *kd*-tree to overcome the computation problem.

We denote $\mathbf{Z}'_t = \mathbf{W}^*\mathbf{Z}_s$ as the embeddings after reconciliation from the source embeddings. In this case, we want to compute similarities between node embeddings from $\mathbf{Z}'_t$ and the target embedding $\mathbf{Z}_t$. We store embeddings from $\mathbf{Z}_t$ into kd-tree. For each node in $\mathbf{Z}'_t$, we can quickly query this tree with its embedding to find the $\alpha \leq n$ closest embeddings from nodes in $\mathbf{Z}_t$. If multiple top alignments are desired, they may be returned in sorted order by their embedding similarity.

7. Experiments and Evaluations

In this section, we conduct experiments and evaluate the performance of our proposed **NAWAL** Framework. We first introduce the experimental setup in Section 7.1. Then, we use three social network datasets to conduct comprehensive experiments and detailed analyses in comparison with existing baseline methods in Section 7.2. Finally, in Section 7.4 we conduct further analysis in a case study to demonstrate its generality.

7.1. Experimental setup

7.1.1. Datasets

To evaluate the efficiency and validity of our proposed framework, three real-world networks are adopted for experimental purposes, including Facebook, FourSquare, and Twitter. The basic statistics of them are presented in Table 2.

Table 2: Statistics of the datasets

Networks	Users	Edges
Facebook	17.359	112.381
FourSquare	17.355	132.208
Twitter	20.024	114.999

Network alignment methods primarily exploit the topology consistency of the network structure to perform alignment as nodes which share the same neighbors often tend to refer to the same entity. Following state-of-the-art alignment algorithms (Zhang & Tong, 2016; Koutra et al., 2013), we study the structural noise by randomly removing edges from real-world datasets. Specifically, for each real network data set with an $\mathbf{A}_s$ adjacency matrix, we build a new network with the $\mathbf{A}_t = \mathbf{P}\mathbf{A}_s\mathbf{P}^T$ adjacency matrix, where $\mathbf{P}$ is a randomly generated permutation matrix with non-zero values that indicate the ground-truths for the network alignment problem. We add structural noise to $\mathbf{A}_t$ by removing edges with probability ranging from 0% to 40% without disassociating any nodes. With experiments requiring imbalance between source and target networks, we remove nodes from $\mathbf{V}_p$ with probability p to create the target network.

7.1.2. Baseline methods

We compare our approach to six state-of-the-art network alignment methods:

Unsupervised Methods. Like our approach, these methods do not use any labelled data, but sometimes rely on hand-crafted features (e.g., username similarity between two social network users) to create prior alignment preference (Zhang & Tong, 2016):

- *IsoRank:* a popular spectral alignment technique which models the similarity of nodes based on the correlation between the degree of their neighbors (Singh et al., 2008).
- *FINAL:* a spectral alignment technique which defines three criteria namely structure similarity, node feature similarity and edge feature similarity to

tackle alignment problems on attributed networks (Zhang & Tong, 2016).

- *REGAL:* a spectral alignment technique which employs low-rank matrix factorization approximation to speed up calculation of alignment matrix (Heimann et al., 2018).

- *UAGA:* a representation learning based technique that which learn the embeddings focusing on high-order proximity using random walks, then uses a GAN-based mapping function to reconcile the learnt embedding spaces (Chen et al., 2019).

Supervised Methods. These methods use labelled links or pre-defined anchor links between the source and target networks as supervision data to train their model:

- *PALE:* a representation learning based technique which learns node embeddings by maximizing the co-occurrence likelihood of vertices, then leverages linear or MLP as the mapping function (Man et al., 2016).

- *DeepLink:* a representation learning based technique which generates the embeddings using the skip-gram model, then uses auto-encoder and MLP to construct the mapping function (Zhou et al., 2018).

- *IONE:* a representation learning based technique which adopts the first-order proximity as input-output context to learn node embeddings, which are used later to generate alignment results. (Liu et al., 2019).

7.1.3. Metrics

In this section, we present similarity metrics to evaluate the performance of the algorithms. These metrics can be separated into two categories: *prediction metric* and *ranking metric*.

Prediction metric. Recent approaches consider network alignment as a binary classification task. We simply employ a setwise metric (Zhang et al., 2014) to define accuracy metric which is calculated by:

$$acc = \frac{\#\{\text{correctly identified node pairs}\}}{\#\{\text{groundtruth node pairs}\}} \tag{14}$$

Ranking metric. Some approaches may provide a top-k ranking list of potential matching nodes rather than only one. The goal is to rank true matching identities as accurate as possible. These metrics may greatly support applications related to information retrieval or recommendation systems so that we also take MAP (Man et al., 2016) and Precision@k (Iofciu et al., 2011) into consideration for evaluating the algorithms.

7.1.4. Settings

Due to the randomness, we run each data set 50 times to compute average results. For our discriminator, we use MLP with two hidden layers of size 2048, and Leaky-ReLU activation functions. The input to the discriminator is corrupted with dropout noise with a rate of 0.1. As suggested by (Goodfellow, 2016), we include a smoothing coefficient s = 0.1 in the discriminator predictions. We use stochastic gradient descent with a batch size of 64, a learning rate of 0.1, and a decay of 0.98 both for the discriminator and W. We divide the learning rate by 2 every time our unsupervised validation criterion decreases. For other algorithms, we tune the parameters to have the best experiment performance. We use alignment accuracy as the main evaluation metric to measure the performance. Besides, we employ some ranking metrics to illustrate the generality of the algorithms. All the experiments are conducted on an AMD 3.0 GHz system with 32 GB of main memory and four GTX Titan X graphic cards.

7.2. Alignment Performance Analysis

To assess the reliability of **NAWAL**, we empirically interpret the effectiveness of our unsupervised approach on several benchmarks and compare it with both state-of-the-art unsupervised and supervised methods. For each learning methods, we conduct a suitable analysis on robustness to structural noise and graph size imbalance.

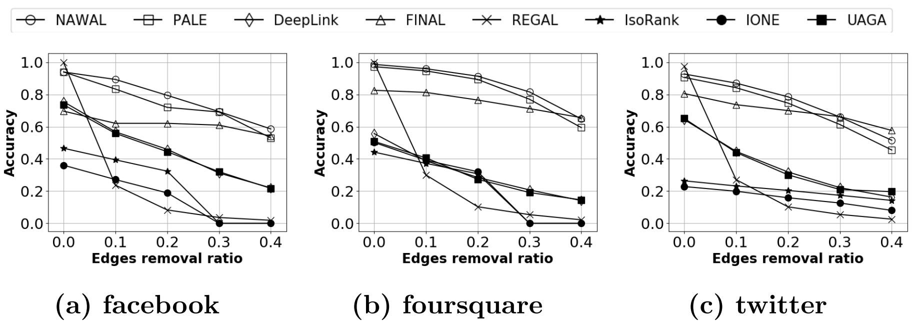


Figure 4: Robustness of algorithms to structural noise.

7.2.1. Robustness to structural noise

Network alignment methods utilize the structure of the network as the main information for the algorithms. Therefore, it is necessary to verify the stability of the model for network structural noise. To stimulate the structural noise, we consider each of the twenty-one experimental settings (i.e.,(3 datasets) x (7 levels of structural noise ranging from 0% to 40%)) as trials and consider what performance trends are likely to generalize (Table 3).

Comparative performance to unsupervised methods. From Fig.4, we can see that the accuracy of all algorithms is monotonically decreasing along with the increasing of the level of structural noise. Based on the results in Fig.4 and Table 3, we can see that the proposed NAWAL model gives better network alignment results than the other unsupervised models in most cases. Our technique achieves 35 - 40% higher accuracy than the second best technique UAGA when the noise level reaches to 0.4 for all the three datasets. This is because our model focus on captures the first-order proximity information (through the loss function) instead of the higher-order proximity as in UAGA, which helps to avoid the inconsistency due to random sampling. REGAL starts with very high performance for a noise-free case, however, its accuracy drops sharply when the level of noise increases. From these results, we can see that the IsoRank method exhibits the worst performance as a trivial method. However, when the level of noise increases (i.e., over 10%), IsoRank perform better than REGAL. This is because REGAL adopts a strict assumption on topology consistency that the two nodes are similar when their neighbor's degree is the same, which makes the model susceptible to a considerable level of structure noise. The performance of the FINAL has less sensitivity for the structural noise but its performance remains at a medium level (from lower than 60% to 80%). In the worst case, FINAL outperforms NAWAL, however, NAWAL is superior to FINAL in most cases. The average accuracy on three datasets of NAWAL model is 84.04%, suggesting performance improvement over IsoRank, REGAL and FINAL models by 57.85%, 42.77% and 13.28%, respectively.

Table 3: Average performance of algorithms to structural noise.

Methods	**Facebook**				**FourSquare**				**Twitter**			
	ACC	**MAP**	**P@5**	**P@10**	**ACC**	**MAP**	**P@5**	**P@10**	**ACC**	**MAP**	**P@5**	**P@10**
Unsupervised methods												
IsoRank	23.69	30.77	39.08	44.33	22.44	29.41	37.31	43.96	20.33	30.59	41.09	51.81
FINAL	61.82	75.33	92.97	94.47	75.49	80.82	86.86	86.88	69.66	76.34	83.99	84.03
REGAL	27.46	32.55	37.19	42.70	29.56	32.99	36.46	39.17	28.61	32.36	36.31	39.56
UAGA	47.32	57.01	68.27	76.71	31.07	42.25	53.07	61.75	35.11	45.16	56.74	66.91
Supervised methods												
PALE	74.32	79.62	85.67	89.12	83.57	87.40	91.91	94.45	71.29	76.96	83.44	87.37
DeepLink	46.29	56.30	67.68	76.95	31.57	41.82	52.93	62.18	35.89	46.01	57.22	66.24
IONE	16.43	22.72	29.42	35.47	24.29	29.64	35.43	40.25	15.85	20.76	24.79	29.88
Our proposed method												
NAWAL	78.17	82.82	88.24	91.23	86.52	89.73	93.52	95.54	75.21	80.26	85.95	89.39

Comparative performance to supervised methods. In this experiment, we use 20% of the number of known anchor links to guide supervised models (PALE, DeepLink). However, without pre-defined anchor links, our proposed method outperforms the existing supervised alignment methods. Overall, we find that the NAWAL and PALE algorithm perform the best, in terms of both average accuracy and structural noise robustness. However, when comparing the results of PALE and NAWAL, we can see that the NAWAL model often works better. The average performance achieved by IONE is generally lower than the other methods. Although both DeepLink and IONE are sensitive to the structural noise factor, DeepLink works better. The average accuracy of the NAWAL model is 84.04%, indicating that it outperforms the IONE model by 61.17%, the DeepLink model by 39.94%, and the PALE model by 3.09%. In order to evaluate

the performance on the ranking metrics, we evaluate different approaches in terms of MAP, top-5 Precision (P@5), and top-10 Precision (P@10) in Table 3.
540 It can be seen that the performance on ranking metrics is basically consistent with the accuracy.

7.2.2. Robustness to graph size imbalance

This experiment evaluates the robustness of all models through another factor of noise, that is the graph size difference between the two networks.

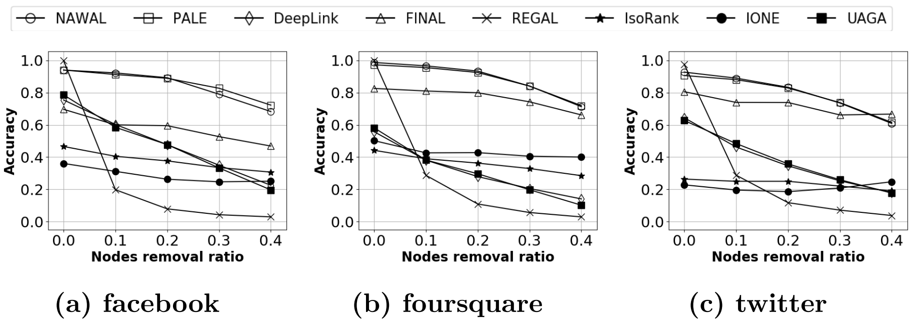


Figure 5: Robustness of algorithms to graph size imbalance.

545 In Fig. 5, we provide the results of our algorithm compared to other state-of-the-art recent algorithms by varying the graph size imbalance factor from 0% to 40% and in Table 4, we provide qualitative results of these frameworks. From the quantitative results, we can see that NAWAL framework performs better than other methods on average.

550 *Comparative performance to unsupervised methods.* From the comparison experiments in Fig.5, it can be seen that representation learning-based methods such as UAGA and REGAL are sensitive in accurately aligning the test nodes when dealing with graph size imbalance factor. This indicates that taking into account the imbalance factor, the latent representations of the source and target
555 networks are harder to align. REGAL suffers a significant drop when the imbalance factor increases. IsoRank shows worse performance than other methods. However, its accuracy is less broken by imbalance factor than REGAL, which results in its accuracy being higher than REGAL when imbalance factor exceeds

10%. UAGA also suffers a considerable accuracy drop of around 40% when the removal ratio reaches up to 0.4 for all datasets. FINAL, matrix-factorization based method, is basically consistent with all datasets and is able to an achieve average result in each case. From the experimental results in Table 4, it can be derived that the average accuracy of NAWAL model is 87.37%, which outperforms the IsoRank, REGAL and FINAL models by 53.11%, 46.34% and 16.50%, respectively.

Table 4: Average performance of algorithms to graph size imbalance.

Methods	**Facebook**				**FourSquare**				**Twitter**			
	ACC	**MAP**	**P@5**	**P@10**	**ACC**	**MAP**	**P@5**	**P@10**	**ACC**	**MAP**	**P@5**	**P@10**
Unsupervised methods												
IsoRank	37.81	49.86	63.97	73.47	36.20	47.93	61.29	72.35	23.51	34.87	46.76	58.96
FINAL	57.77	72.44	92.09	95.04	76.83	81.51	86.64	86.65	72.27	78.33	85.02	85.04
REGAL	26.96	32.27	36.90	43.20	29.64	33.14	36.47	39.43	29.84	33.92	38.26	41.56
UAGA	49.89	57.86	73.28	81.64	32.49	42.34	51.44	61.98	38.69	50.41	60.48	71.17
Supervised methods												
PALE	85.85	89.43	93.74	95.56	88.28	91.43	95.29	97.00	79.40	84.35	90.24	93.37
DeepLink	48.22	58.89	71.43	80.78	31.30	41.61	52.73	62.01	37.74	48.45	60.46	69.99
IONE	28.68	39.28	50.76	60.67	43.27	52.13	61.96	69.39	21.34	27.67	33.31	39.75
Our proposed methods												
NAWAL	84.58	88.31	92.63	94.76	88.83	91.69	95.14	96.82	79.93	84.57	90.07	92.92

Comparative performance to supervised methods. From the results in Fig.5, it can be concluded that the NAWAL method and PALE method always perform better than the DeepLink. Besides, the NAWAL model is on par with or outperforms PALE in most cases. The performance of DeepLink decreases gradually when the imbalance factor increases. In average, on three datasets, for NAWAL, the percentages of speedup contributed by IONE, DeepLink and PALE are around 55.00%, 42.56% and 0.43%, respectively.

7.2.3. Robustness to scalability

In this experiment, we investigate the scalability of the network alignment techniques by studying the computation time each technique needed to process the large-size input networks. We try our best to apply a broad range to the setting, with the number of nodes ranging from 1,000 to 1,000,000 nodes. As the size of the input networks cannot be manipulated using the real-world dataset, we use the Small-world network generative Watts & Strogatz (1998) model to synthesize the input networks. Because computation time increases exponentially with the growth of the input network size, we apply logarithmic scale. We set the memory limit to 24GiB as we want to consider gpu and cpu memory in a fair manner.

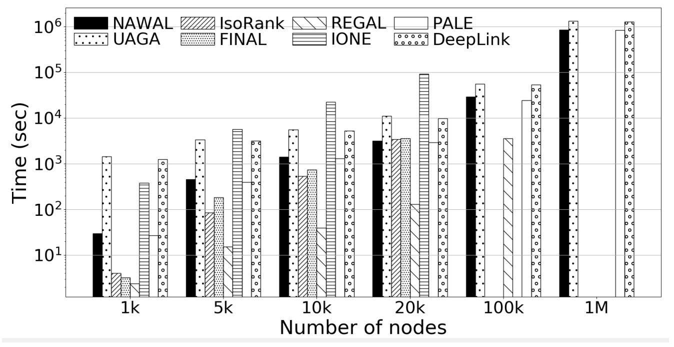


Figure 6: Effect of graph size on running time.

The result of the experiment is shown in Figure 6. In general, the computation time of the techniques significantly increase with the growth of the network size. In more details, the computation of matrix factorization based methods such as IsoRank, REGAL and FINAL are relative lower than that of representation based methods when the size of input networks is small, less than 10,000 nodes. However, this figure of such methods soars up when the input size become larger, and cannot be traced when the network node size reaches 1,000,000 because of the explosion of memory usage. On the other hands, all the representation learning based methods like NAWAL, UAGA, PALE and DeepLink all show good scalability, as they are all able to handle when the network size goes high. Our technique has quite similar computation time with PALE, and better than that of UAGA and DeepLink with around 500,000 seconds faster when the network size reaches 1,000,000. This is because our technique as well as PALE use direct neighborhoodship instead of random sampling, which is a costly process in terms of running time.

7.3. Ablation tests

In this section, we study the importance of each proposed components in our framework by comparing the performance of our final model to several variants. The details of the variants are as follows.

- NAWAL-1: employs the linear mapping function (Man et al., 2016) to reconcile the learnt embedding spaces.
- NAWAL-2: uses the multi-layer perceptron (Man et al., 2016) to reconcile the learnt embedding spaces.
- NAWAL-3: leverages a sophisticate AutoEncoder mapper (Zhou et al., 2018) to unifies the learnt representation spaces.
- NAWAL-4: has the similar pipeline as the final model, except not using the refinement step described in Section. 5.2.

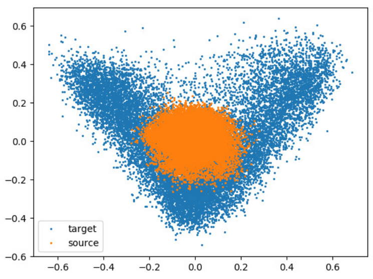
Table 5 presents the result with only important metrics and datasets due to space limitation. It can be seen that our original model *NAWAL* outperforms other variants. In particular, *NAWAL* model achieves higher accuracy and P@5 (around 2-3%) than *NAWAL-1*, *NAWAL-2* and *NAWAL-3* for all datasets. This proves the superiority of the GAN-based mapping function to the other function, given that all these reconciliations are performed on the same representation learning technique. Also, *NAWAL* in general performs better than *NAWAL-4*, which confirms the need of refinement step to enhance the quality of self-supervised ground-truth.

Table 5: Ablation test

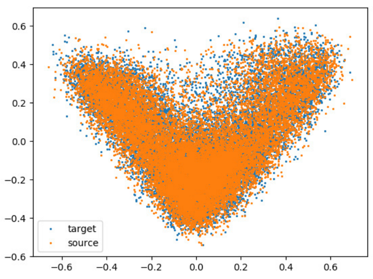
Dataset	**Metric**	**NAWAL**	**NAWAL-1**	**NAWAL-2**	**NAWAL-3**	**NAWAL-4**
Foursquare	Accuracy	**0.9603**	0.9468	0.9446	0.9277	0.9548
	P@5	**0.9842**	0.9794	0.9698	0.9541	0.9828
Facebook	Accuracy	**0.9228**	0.9127	0.9120	0.9012	0.9053
	P@5	**0.9703**	0.9665	0.9589	0.9328	0.9316

7.4. Case studies

In this section, we present some visualization results (best viewed in color) to intuitively demonstrate the effect of the embedding reconcilement step (Step 2 of NAWAL Framework). We leverage the PCA algorithm to visualize the data distributions. The figures show the embeddings before (Fig.7-a, Fig.8-a, and Fig.9-a) and after (Fig.7-b, Fig.8-b, and Fig.9-b) being reconciled into a common space. The mapped source embeddings ($\mathbf{WZ}_s$) is in orange, and the target embeddings ($\mathbf{Z}_t$) is in blue. There is a strong correspondence between the two aligned data distributions which leads to the success of the whole network alignment process in terms of both noise robustness and accuracy.

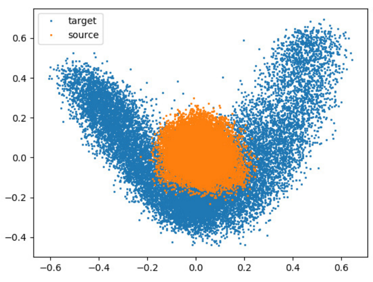


(a) before

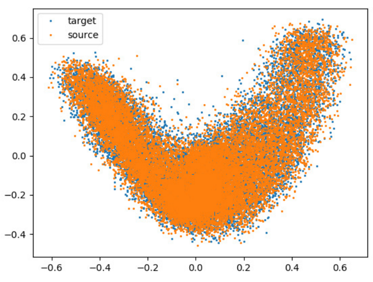


(b) after

Figure 7: The effect of the embedding reconcilement step on the Facebook dataset.

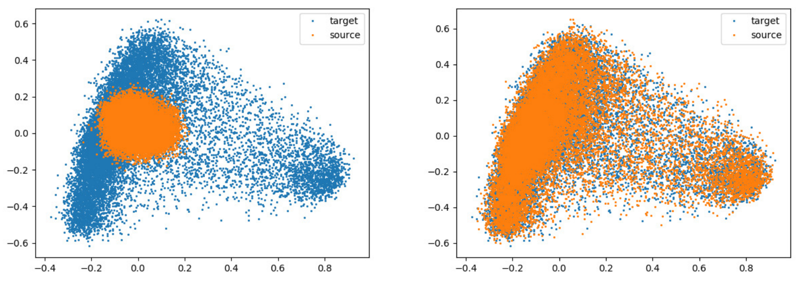


(a) before



(b) after

Figure 8: The effect of the embedding reconcilement step on the FourSquare dataset.



(a) before **(b) after**

Figure 9: The effect of the embedding reconcilement step on the Twitter dataset.

8. Related Work

Related work mainly includes two parts. The first part reviews various network embedding methods which can be exploited as a pre-processing step for the network alignment framework. The second part is on network alignment which is the problem we focus on this study.

8.1. Network Embedding

Our work is related to the problem of network embedding as the node representations of the source and the target networks are learned independently at the first step of NAWAL framework. Network embedding methods aim to learn a low-dimensional representation of nodes in the network (Cai et al., 2018). Many network embedding approaches have been proposed in the past decades. Classical graph embedding algorithms such as IsoMap (Tenenbaum et al., 2000), Laplacian Eigenmaps (Belkin & Niyogi, 2003) are not applicable for embedding large-scale networks. These techniques construct a unique embedding for each node directly and efficiently extract information from a network with a low computational cost. However, due to their limited representation abilities, these methods cannot learn a highly non-linear mapping when encoding the network structure from a high dimensional to a low-dimensional space.

Recent methods that model the graph-structured input by neural networks have been proposed such as LINE (Tang et al., 2015), DeepWalk (Perozzi &

Skiena, 2014), SDNE (Wang et al., 2016), node2vec (Grover & Leskovec, 2016), and GraphSAGE (Hamilton et al., 2017). These methods have a high ability to model a non-linear transformation of network structure. They learn node embeddings by optimizing the first-order or higher-order node similarity. The embedding algorithm used by NAWAL is inline with the first-order node similarity methods. We do not consider the methods which maximize the higher-order similarity as they can raise computational and time expense due to the sampling step.

8.2. Network Alignment

Network alignment problem has drawn considerable research interests in recent years. This problem has been widely applied in many application areas such as database schema matching (Melnik et al., 2002), data mining (Bayati et al., 2009), social network (Feng et al., 2018), computer vision (Yang et al., 2018a) to bioinformatics (Hashemifar & Xu, 2014; Singh et al., 2008). Recent approaches can be categorized into *unsupervised* and *supervised* alignment methods.

The unsupervised methods aim to tackle the problem of network alignment in a general manner and hence, they did not exploit any anchor links. Most of them compute the network alignment by adopting a matrix factorization. Although these approaches are quite straightforward, it has been proven to solve the problem effectively. One famous algorithm is IsoRank (Singh et al., 2008), inspired by PageRank (Xing & Ghorbani, 2004), which constructs an eigenvalue problem for every pair of input networks with the assumption that the protein in a PPI network is compatible with the protein in another network if the source node's neighbors match the neighbors of the target node. NetAlign (Bayati et al., 2009) models the alignment problem as NP-hard combinatorial optimization problem and approximates it by applying the belief propagation heuristic. BigAlign (Koutra et al., 2013) then reformulates bipartite network alignment as a new optimization problem and proposes a gradient-descent-based algorithm to solve it effectively. Again, on an optimization-based perspective, FINAL (Zhang & Tong, 2016) also develop an algorithm to find the optimal alignment result by preserving network consistency principle. User alignment, also called social identity linkage (Liu et al., 2014), anchor link prediction (Kong et al., 2013), cross-platform identification of anonymous identical users (Zhou et al., 2015) in some literatures, aims at identifying one user's accounts across social networks. Due to the rich featured nodes of social networks, some aligners take both user profiles (i.e., username, locations, posts) and structural similarity into account to compute the alignment matrix score (Liu et al., 2013). CoLink (Zhong et al., 2018) incorporates a sequence-to-sequence attributed-based model, and a heuristic relationship-based model to solve the problem of user alignment across social networks. Using an approximation of low-rank matrix factorization and representation learning, REGAL (Heimann et al., 2018) further accelerates the network alignment process.

Inspired by network embedding techniques (Perozzi & Skiena, 2014; Grover & Leskovec, 2016; Liu et al., 2018; Ji et al., 2018; Hamilton et al., 2017; Pham & Do, 2019), many supervised alignment approaches have been proposed under the theme of embedding and mapping approaches to deal with large-scale networks. Those approaches aim to induce network alignment work by independently learning the embeddings in each network using single network structure, and then learning a mapping from one embedding space into the other based on anchor links. The first of such methods is PALE (Man et al., 2016) which learns nodes embedding by maximizing the co-occurrence likelihood of nodes then applies linear or multilayer perceptron (MLP) as mapping function. DeepLink (Zhou et al., 2018) employs unbiased random walk to generate embeddings using skip-gram then using auto-encoder and MLP to construct mapping function. UAGA Chen et al. (2019) leverages the same embedding technique as DeepLink, but adopts generative adversarial network (GAN) based mapping function to reconcile the learnt spaces in an unsupervised manner. Mego2vec (Zhang et al., 2018) incorporated both attribute embeddings and structural embeddings into a convolutional neural network to train the model with alignment labels. IONE (Liu et al., 2019) then uses the same mapping function as PALE but its embedding process is more complicated as it takes into account second-order node similarity.

Table 6: Characteristics of different network alignment techniques

Technique	Feature-based	Learning Type
IsoRank (Singh et al., 2008)	No	Unsupervised
BigAlign (Koutra et al., 2013)	Yes	Unsupervised
FINAL (Zhang & Tong, 2016)	Yes	Unsupervised
CoLink (Zhong et al., 2018)	Yes	Weakly-supervised
REGAL (Heimann et al., 2018)	Yes	Unsupervised
PALE (Man et al., 2016)	No	Supervised
Meg2vec (Zhang et al., 2018)	Yes	Supervised
DeepLink (Zhou et al., 2018)	No	Supervised
IONE (Liu et al., 2019)	No	Supervised
UAGA (Chen et al., 2019)	No	Unsupervised

A summary comparison of existing network alignment techniques is shown in Table 6. Whereas these methods put in a solid performance in some large datasets, they rely only on labeled data where the cost of generating them for a new machine learning task is often an obstacle for applying learning methods. To bridge this gap, in our end-to-end setting, we propose a GAN-based approach to bridge the gap so that our model is able to learn a mapping between the source and target network in the situation when the network data are either fully unlabeled. While our model and UAGA both use GAN-based reconciliation, our technique focus on capturing the first-order proximity by using the the direct neighborhoodship instead of using random sampling. This is because direct neighborhoodship is simple yet effective context as it captures the essential information of network structure, while significantly reduces the computation effort and time expense due to the sampling step. Besides, the inconsistency due to random sampling often amplifies the structural noises. Also, we carefully design the refinement step that employs the cross-domain similarity local scaling to enhance the quality of the self-supervised ground-truth iteratively. The cross-domain similarity local scaling favors the less noisy and localized first-order proximity information encoded in our representation learning model.

9. Conclusions

This paper proposes *NAWAL*, an end-to-end, unsupervised framework for network alignment, which is a fundamental step for emerging intelligence and expert systems such as multiple social networks analysis or cross-lingual knowledge graph reconciliation. The contributions of this paper can be seen in methodological and empirical viewpoints.

In methodological perspective, NAWAL goes beyond the existing methods in two aspects. First, it aims to propose an unsupervised network alignment approach based on the adversarial training framework, which learns the light-weight latent node features to scale the alignment. Second. the alignment is an iterative self-supervised process of refining the matching (similarity) matrix from the currently defined anchor links (top matching pairs from the previous matrix) until the optimisation does not improve. In practice, our approach can be seamlessly integrated with existing supervised methods, which leverages domain expert or prior knowledge as guidance.

In empirical perspective, NAWAL improves on the existing methods on both scalable and autonomous aspects. Experiments on best-practice benchmarking datasets parameterised settings, and state-of-the-art alignment methods verify the efficiency and effectiveness of our method on real-world and large-scale social networks, including Facebook, FourSquare, and Twitter. Our approach generally outperforms the baselines by a margin of over 13% accuracy in average and by without using any prior knowledge.

In the future, this work can be extended in several directions. First, NAWAL incorporates the first-order structural information for embedding step - which is efficient and a natural choice by existing works. However, it shall be interesting to explore some higher-order network representations such as Graph Convolutional Network (Zhang et al., 2019). Second, node attribute should be aggregate to increase the knowledge representation capacity as it plays a critical role in some network domains. For example, in a social network, a user who engages in multiple social platforms may have a common name, which is useful indicative information for the reconciliation step. Third, we shall extend this alignment technique to knowledge graph alignment, which is useful to a variety of applications (e.g., question-answering (Bakhshi et al., 2020), ontology alignment (Karimi & Kamandi, 2019)). Finally, we plan to exploit some parallel and distributed techniques to improve the efficiency to support for a wide range of real-time downstream applications.

Acknowledgement

The research is supported by Vingroup Innovation Foundation (VINIF) in project code VINIF.2019.DA01.

References

Bakhshi, M., Nematbakhsh, M., Mohsenzadeh, M., & Rahmani, A. M. (2020). Data-driven construction of sparql queries by approximate question graph alignment in question answering over knowledge graphs. *Expert Systems with Applications*, *146*, 113205.

Bayati, M., Gerritsen, M., Gleich, D. F., Saberi, A., & Wang, Y. (2009). Algorithms for large, sparse network alignment problems. In *2009 Ninth IEEE International Conference on Data Mining* (pp. 705–710). IEEE.

Belkin, M., & Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, *15*, 1373–1396.

Bhatia, N. et al. (2010). Survey of nearest neighbor techniques. *arXiv preprint arXiv:1007.0085*, .

Cai, H., Zheng, V. W., & Chang, K. C.-C. (2018). A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, *30*, 1616–1637.

Cao, X., Zhang, W., & Yu, Y. (2018). A bootstrapping framework with interactive information modeling for network alignment. *IEEE Access*, *6*, 13685–13696.

Chen, C., Xie, W., Xu, T., Rong, Y., Huang, W., Ding, X., Huang, Y., & Huang, J. (2019). Unsupervised adversarial graph alignment with graph embedding. *arXiv preprint arXiv:1907.00544*, .

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., & Usunier, N. (2017). Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (pp. 854–863). JMLR. org.

Conneau, A., Lample, G., Ranzato, M., Denoyer, L., & Jégou, H. (2017). Word translation without parallel data. *CoRR*, *abs/1710.04087*.

Du, Y., Guo, W., Liu, J., & Yao, C. (2019). Classification by multi-semantic meta path and active weight learning in heterogeneous information networks. *Expert Systems with Applications*, *123*, 227–236.

Feng, S., Shen, D., Nie, T., Kou, Y., He, J., & Yu, G. (2018). Inferring anchor links based on social network structure. *IEEE Access*, *6*, 17340–17353.

Francisquini, R., Rosset, V., & Nascimento, M. C. (2017). Ga-lp: A genetic algorithm based on label propagation to detect communities in directed networks. *Expert Systems with Applications*, *74*, 127–138.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., & Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, *17*, 2096–2030.

Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, .

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 27* (pp. 2672–2680). Curran Associates, Inc. URL: `http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf`.

Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *KDD* (pp. 855–864). ACM.

Gupta, M., Kumar, P., & Bhasker, B. (2017). Heteclass: A meta-path based framework for transductive classification of objects in heterogeneous information networks. *Expert Systems with Applications*, *68*, 106–122.

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30* (pp. 1024–1034).

Han, T., Tian, Y.-C., Lan, Y., Li, F., & Xiao, L. (2018). Revealing the densest communities of social networks efficiently through intelligent data space reduction. *Expert Systems with Applications*, *94*, 70–80.

Hashemifar, S., & Xu, J. (2014). Hubalign: An accurate and efficient method for global alignment of protein-protein interaction networks. *Bioinformatics (Oxford, England)*, .

Heimann, M., Shen, H., Safavi, T., & Koutra, D. (2018). Regal: Representation learning-based graph alignment. In *CIKM* (pp. 117–126).

Iofciu, T., Fankhauser, P., Abel, F., & Bischoff, K. (2011). Identifying users across social tagging systems. In *Fifth International AAAI Conference on Weblogs and Social Media*.

Ji, H., Shi, C., & Wang, B. (2018). Attention based meta path fusion for heterogeneous information network embedding. In *Pacific Rim International Conference on Artificial Intelligence* (pp. 348–360). Springer.

Karimi, H., & Kamandi, A. (2019). A learning-based ontology alignment approach using inductive logic programming. *Expert Systems with Applications*, *125*, 412–424.

Kollias, G., Mohammadi, S., & Grama, A. (2012). Network similarity decomposition (nsd): A fast and scalable approach to network alignment. *TKDE*, *24*, 2232–2243.

Kong, X., Zhang, J., & Yu, P. S. (2013). Inferring anchor links across multiple heterogeneous social networks. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management* (pp. 179–188). ACM.

Koutra, D., Tong, H., & Lubensky, D. (2013). Big-align: Fast bipartite graph alignment. In *ICDM* (pp. 389–398).

Liu, J., Zhang, F., Song, X., Song, Y.-I., Lin, C.-Y., & Hon, H.-W. (2013). What's in a name?: an unsupervised approach to link users across communities. In *Proceedings of the sixth ACM international conference on Web search and data mining* (pp. 495–504). ACM.

Liu, L., Li, X., Cheung, W., & Liao, L. (2019). Structural representation learning for user alignment across social networks. *IEEE Transactions on Knowledge and Data Engineering*, .

Liu, S., Wang, S., Zhu, F., Zhang, J., & Krishnan, R. (2014). Hydra: Large-scale social identity linkage via heterogeneous behavior modeling. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data* (pp. 51–62). ACM.

Liu, X., Kertkeidkachorn, N., Murata, T., Kim, K.-S., Leblay, J., & Lynden, S. (2018). Network embedding based on a quasi-local similarity measure. In *Pacific Rim International Conference on Artificial Intelligence* (pp. 429–440). Springer.

Man, T., Shen, H., Liu, S., Jin, X., & Cheng, X. (2016). Predict anchor links across social networks via an embedding approach. In *IJCAI* (pp. 1823–1829). volume 16.

Melnik, S., Garcia-Molina, H., & Rahm, E. (2002). Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In *ICDE* (pp. 117–128).

Mikolov, T., Le, Q. V., & Sutskever, I. (2013a). Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, .

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *NIPS* (pp. 3111–3119). Curran Associates, Inc.

Nisha, C., & Mohan, A. (2019). A social recommender system using deep architecture and network embedding. *Applied Intelligence*, *49*, 1937–1953.

Pandey, B., Bhanodia, P. K., Khamparia, A., & Pandey, D. K. (2019). A comprehensive survey of edge prediction in social networks: Techniques, parameters and challenges. *Expert Systems with Applications*, .

Perozzi, R., B.; Al-Rfou, & Skiena (2014). Deepwalk: Online learning of social representations. In *KDD*.

Pham, P., & Do, P. (2019). W-metapath2vec: The topic-driven meta-path-based model for large-scaled content-based heterogeneous information network representation learning. *Expert Systems with Applications*, *123*, 328–344.

Schönemann, P. H. (1966). A generalized solution of the orthogonal procrustes problem. *Psychometrika*, *31*, 1–10.

Singh, R., Xu, J., & Berger, B. (2008). Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences*, *105*, 12763–12768.

Smith, S. L., Turban, D. H., Hamblin, S., & Hammerla, N. Y. (2017). Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*, .

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web* (pp. 1067–1077). International World Wide Web Conferences Steering Committee.

Tenenbaum, J. B., De Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *science*, *290*, 2319–2323.

Toan, N. T., Cong, P. T., Tam, N. T., Hung, N. Q. V., & Stantic, B. (2018). Diversifying group recommendation. *IEEE Access*, *6*, 17776–17786.

Wang, D., Cui, P., & Zhu, W. (2016). Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1225–1234). ACM.

Wang, Z., Liang, J., & Li, R. (2018). Exploiting user-to-user topic inclusion degree for link prediction in social-information networks. *Expert Systems with Applications*, *108*, 143–158.

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, *393*, 440–442. URL: `http://dx.doi.org/10.1038/30918`.

Xing, W., & Ghorbani, A. (2004). Weighted pagerank algorithm. In *Proceedings. Second Annual Conference on Communication Networks and Services Research, 2004.* (pp. 305–314). IEEE.

Yang, H., Song, D., & Liao, L. (2018a). Image captioning with relational knowledge. In *Pacific Rim International Conference on Artificial Intelligence* (pp. 378–386). Springer.

Yang, J., Zhang, M., Shen, K. N., Ju, X., & Guo, X. (2018b). Structural correlation between communities and core-periphery structures in social networks: Evidence from twitter data. *Expert Systems with Applications*, *111*, 91–99.

Zhang, H., Kan, M.-Y., Liu, Y., & Ma, S. (2014). Online social network profile linkage. In *Asia Information Retrieval Symposium* (pp. 197–208). Springer.

Zhang, J., Chen, B., Wang, X., Chen, H., Li, C., Jin, F., Song, G., & Zhang, Y. (2018). Mego2vec: Embedding matched ego networks for user alignment across social networks. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (pp. 327–336). ACM.

Zhang, J., & Philip, S. Y. (2015). Multiple anonymized social networks alignment. In *2015 IEEE International Conference on Data Mining* (pp. 599–608). IEEE.

Zhang, S., & Tong, H. (2016). Final: Fast attributed network alignment. In *KDD* (pp. 1345–1354).

Zhang, S., Tong, H., Xu, J., & Maciejewski, R. (2019). Graph convolutional networks: a comprehensive review. *Computational Social Networks*, *6*, 11.

Zhong, Z., Cao, Y., Guo, M., & Nie, Z. (2018). Colink: An unsupervised framework for user identity linkage. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhou, F., Liu, L., Zhang, K., Trajcevski, G., Wu, J., & Zhong, T. (2018). Deeplink: A deep learning approach for user identity linkage. In *INFOCOM* (pp. 1313–1321).

Zhou, X., Liang, X., Zhang, H., & Ma, Y. (2015). Cross-platform identification of anonymous identical users in multiple social media networks. *IEEE transactions on knowledge and data engineering*, *28*, 411–424.

***Highlights (for review)**

- Our approach is scalable to real-world online social networks
- Our method is robust to structural noises up to 40%
- Our model outperforms unsupervised methods by over 13% accuracy
- Our model is cheaper than supervised methods in not using any prior knowledge
- Our performance is robust and maintains 90% accuracy even if data is 10% sparser

***Credit Author Statement**

Nguyen Thanh Toan: Conceptualization, Methodology, Formal analysis, Validation, Writing - Original Draft. **Pham Minh Tam**: Resources, Data Curation, Visualization. **Nguyen Thanh Tam**: Methodology, Formal analysis, Writing - Review & Editing. **Huynh Thanh Trung**: Methodology, Validation. **Tong Van Vinh**: Methodology, Validation. **Nguyen Quoc Viet Hung**: Conceptualization, Writing - Review & Editing. **Quan Thanh Tho**: Supervision.

***Conflict of Interest**

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: