



Network Alignment by Representation Learning on Structure and Attribute

Thanh Trung Huynh¹(✉), Van Vinh Tong², Chi Thang Duong³,
Thang Huynh Quyet², Quoc Viet Hung Nguyen¹, and Abdul Sattar¹

¹ Griffith University, Brisbane, Australia

{h.thanhtrung, henry.nguyen, a.sattar}@griffith.edu.au

² Hanoi University of Science and Technology, Hanoi, Vietnam

thanghq@soict.hust.edu.vn

³ EPFL, Lausanne, Switzerland

thang.duong@epfl.ch

Abstract. Network alignment is the task of recognizing similar network nodes across different networks, which has many applications in various domains. As traditional network alignment methods based on matrix factorization do not scale to large graphs, a variety of representation learning based approaches has been proposed recently. However, these techniques tend to focus on topology consistency between two networks while ignoring other valuable information (e.g. network nodes attribute), which makes them susceptible to structural changes. To alleviate this problem, we propose RAN, a representation-based network alignment model that couples both structure and node attribute information. Our framework first constructs multi-layer networks to represent topology and node attribute information, then computes the alignment result by learning the node embeddings for source and target network. The experimental results show that our method is able to outperform other techniques significantly even on large datasets.

Keywords: Network alignment · Network embedding

1 Introduction

Networks are natural but powerful structures that capture relationships between different entities in many domains, such as social networks, citation networks, bioinformatics, chemistry. In many applications that involves multiple networks analysis, network alignment, the task of recognizing node correspondence across different networks, plays an important role. For example, by detecting accounts from the same user in different social networks, information of that user in one site can be exploited to perform better downstream functions (e.g. friend suggestion or content recommendation) in the other site [20]. In computer vision, network alignment helps to match images without human supervision [19]. In bioinformatics, analysis of protein-protein interactions networks across species makes remarkable improvement for gene prioritization [7].

Despite its ubiquity, network alignment is challenging as it is an NP-hard problem [3]. There have been several works aiming to deal with this problem using matrix factorization to directly achieve the alignment result, such as Iso-Rank [17], NetAlign [3], UniAlign [10], FINAL [20], REGAL [8]. However, these methods fail to deal with large networks since their computation time grows exponentially with network size.

In order to handle large networks, several alignment techniques [11, 12, 21] attempt to integrate latent representation learning [5, 6, 15] to make their models scalable. Although representation based methods can handle large-scale datasets, most of them focus only on topology consistency, which states that neighborhood relationship between any two nodes is maintained across different networks. However, this assumption can be incorrect in many contexts, for example, a person might have more connections in one social network (e.g. Facebook, Twitter) than others (e.g. LinkedIn, MySpace). When this assumption does not hold, the performance of these methods is adversely affected.

To alleviate this problem, we propose a representation-based network alignment model that couples both topology and node attribute information to enhance the result. Our framework first converts the given networks to multi-layered networks to represent topology and node attribute information, then retrieves the alignment result through learning the node embeddings for source and target network. The experimental results show that our method outperform other techniques significantly even on large datasets.

Our contributions may be stated as follows:

- We formulate the attributed network alignment problem which takes into account the alignment of multiple networks where node attribute information is available.
- We propose a representation-based alignment on attributed network framework (RAN) which identifies node alignments by learning node embeddings for each network and obtaining node pairs having similar embedding across networks after reconciling the embedding spaces.
- Within our framework, we transform the original attributed network into a multi-layer graph to represent both topology and attribute neighborhood and use these information simultaneously in representation learning stage. To the best of our knowledge, we are the first to do so to tackle network alignment problem.
- Experiments on real-world graphs show that our model outperforms state-of-the-art network alignment methods and more robust to structural and attribute noise.

Our paper is organized as follows. Section 2 introduces some necessary preliminaries to our work. Section 3 provides an overview of our approach and the attributed network alignment problem. Section 4 discusses why and how to transform the original networks to multi-layer graphs while Sect. 5 gives details of representation learning strategy with random walk on generated networks. Section 6 describes how to reconcile the embedding spaces and achieve the alignment result. The empirical results are shown in Sect. 7 while Sect. 8 surveys the related works and Sect. 9 concludes the paper.

2 Background

Attributed Network. Network (or graph, we use these terms interchangeably in this paper) is an omnipresent data structure appearing in applications in various fields such as computer science, social science, biology. Basically, a network is presented by a pair $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ comprising a set \mathbf{V} of vertices together with a set \mathbf{E} of edges. However, in many applications, network nodes associate with valuable information called node attributes. Node attributes can be presented as a matrix $\mathbf{N} \in \mathbb{R}^{n \times d}$ with n is the number of node in the network and d is the size of feature vector; each node $v \in \mathbf{V}$ associates with a feature vector $\mathbf{N}(v) \in \mathbb{R}^d$. An network combining with node attribute can now be presented as a triplet $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{N})$.

Network Embedding. Network embedding is an emerging technique in graph analysis due to the ubiquity of large-scale networks in real-world applications. The goal of network embedding is to map network nodes to a low-dimensional embedding space so that similarity of any two nodes reflects their similarity in the original network. Given a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, learning node embeddings requires three main steps:

- Define an encoder $\phi : \mathbf{V} \rightarrow \mathbb{R}^{n \times d}$ associates each network node to a low-dimensional vectorial representation.
- Define an similarity function $\mathbf{sim} : \mathbf{V} \times \mathbf{V} \rightarrow \mathbb{R}$ in the original network that measures the relation between any two nodes.
- Optimize the parameters of the encoder so that similarity function \mathbf{sim} of two nodes approximates the distance between their representation vectors:

3 Model and Approach

3.1 Model

Network alignment is the task of identifying corresponding nodes across two different networks. In this work, we address attributed networks as we use both topology and nodes' feature of the networks to enhance the performance of our proposed network alignment model. Given two attributed 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:

Problem 1. *Given two attributed networks $\mathbf{G}_s = (\mathbf{V}_s, \mathbf{E}_s, \mathbf{N}_s)$ and $\mathbf{G}_t = (\mathbf{V}_t, \mathbf{E}_t, \mathbf{N}_t)$ where $\mathbf{V}_s, \mathbf{V}_t$ are sets of nodes, $\mathbf{E}_s, \mathbf{E}_t$ are sets of edges and $\mathbf{N}_s, \mathbf{N}_t$ are node attribute matrices, 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 [8,9] 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 $\mathbf{M}(u, v) = 0$.

3.2 Approach

Solving the delineated problem is a complex task. Real-world networks often have significant amount of nodes, therefore constructing the potential matching between two networks can be costly, requires the solution method being time efficient and scalable. Besides, in many cases, networks contain structural noises, for example two nodes are neighbors in one network do not always maintain the neighborhood relation in other network. In such case, using topology information alone is insufficient and can mislead to poor alignment result. In our work, therefore, we study an alignment approach that integrating attribute of network nodes into multi-layer network structure.

Figure 1 presents an overview of our model. We start by transforming each given source and target network into a two-layer network. The first layer, namely *structure layer*, corresponds to the structure of the origin network without any node attribute information. The second layer, named *attribute layer*, is constructed by generating pseudo nodes corresponding to nodes feature, called *attribute node*. The intuition behind creating this layer is to capture the relationship between network nodes based on feature through forming links between their attribute nodes. Attribute node of one node also links to its node in structure layer to represent the relationship between topology and feature characteristic of that node. Designing this two-layer network allows us to unify the structure and attribute property of the network and leverage node feature information to facilitate the process of generating nodes representation.

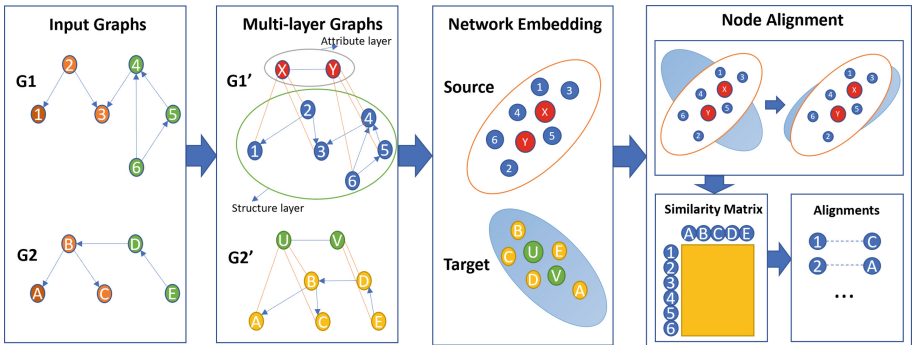


Fig. 1. Overview of RAN model

To learn latent representations of network nodes, we employ a biased random walk and language modeling technique approach. The embedding approach exploits properties of the multi-layer network by adopting random paths over the two layers as nodes' context, enriches the context with both topology and attribute information. Then, the popular skip-gram model is applied on the random walks to learn nodes' embedding by considering walks as the equivalent of sentences in language modeling.

After the embedding step, each network is presented by a low-dimensional space which reflects the characteristic of network nodes through distance: similarity in the embedding space approximates similarity in the original network, which helps to identify the nodes across source and target networks. Because latent space of source and target networks are independently learned, a mapping function is produced in the next step to map the spaces of source and target network into a shared space where the alignment result can be achieved by mapping pairs of nodes that have close embeddings.

Following this structure, our model requires the realization of the following steps:

Multi-layer Network Construction. This step extends each given network to a two-layer network that represents its structure and attribute information. The structure layer maintains the original network topology while the attribute layer captures the attribute neighborhood relationship of network nodes. To construct the attribute layer, first we group the node feature vectors into k clusters, then we generate the attribute nodes presenting each cluster. The attribute node of clusters having close centroids will be linked together to capture nodes feature similarity. Using clustering result makes our model more robust to small changes in attribute introduced by noise and reduces the computation time needed. The detailed process of constructing structure attribute multi-layer networks is described in Sect. 4.

Network Embedding. In this stage, low-dimensional latent spaces are created based on the networks produced in the last step. We want to leverage both topology and attribute information to learn latent representations for network nodes so that more information are used to identify and distinguish the nodes across the networks. As a result, random walks are generated over the two layers and contains both attribute node and structure node. Then, skip-gram model with negative sampling is employed on the random walks to learn embedding for the nodes. The detail of this step is described in Sect. 5.

Node Alignment. As the vectorial embedding spaces for the source and target networks are obtained independently, the embeddings might belong to different and incomparable vector spaces. Therefore, we produce a mapping function that reconcile the spaces by mapping one embedding space to the other. We adopt a supervised algorithm for learning the mapping function inspired by cross-language dictionary building. After being mapped to the common space, nodes from source and target network that have similar representation will be chosen to form the alignment pairs. Node alignment is described in Sect. 6.

4 Multi-layer Network

First we discuss the preprocessing step that convert each source and target network to a multi-layered network. Given the attributed network $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{N})$, we divide the network into two layers. The first layer corresponds to the structure, and the second layer is for the content of the network.

4.1 Structure Layer

This layer is designed to capture the topology information of the original network. To this purpose, we replicate the topology of the original network and discards nodes attribute information in this layer. Mathematically, given the input network $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{N})$, the structure layer is a graph $\mathbf{G}_{st} = (\mathbf{V}_{st}, \mathbf{E}_{st})$, with $\mathbf{V}_{st} = \mathbf{V}$ and $\mathbf{E}_{st} = \mathbf{E}$. There are some reasons for constructing the structure layer this way. First, this helps to avoid losing topology information because any two neighbor nodes in the original network will be connected in the structure layer of the new network. Otherwise, if any two nodes lose their neighborhood relationship during the transformation, they will become undesirable structure noise and consequently adversely affects the final performance of the model due to the homophily principle assumption. Secondly, separating topology and attribute information in two different layers allows us to manage and unify these information easier. While similarity in topology between nodes is presented in this layer, the relation related to attribute will be specified in the attribute layer.

4.2 Attribute Layer

This layer is a directed graph which captures the similarity between nodes in terms of their respective features. Constructing the layer requires two sub-steps:

Creating Attribute Nodes. In this sub-step, we generate pseudo node to represent features of the original network nodes. To prevent the explosion in the amount of pseudo nodes that are generated and taken into account, we bin node features together into k groups, with $k \ll n$, then create one *attribute node* for each group. The grouping strategy depends on the type of node attributes. For categorical node attributes where node feature vectors are in the form of one-hot vectors, we propose using the number of category features as k . For multi-categorical or real-valued node attributed, a variety of popular clustering algorithm can be employed accordingly (K-mean algorithm is used in our work). It is worth noting that this process not only reduces the amount of attribute node, it also makes our model more resistant to small changes in feature vectors introduced by noise as their representing *attribute node* remain the same.

Connecting Attribute Nodes. After forming the attribute nodes, we assign the edges between these nodes to simulate the attribute similarity between network nodes. For each attribute node $v_i \in \mathbf{V}$, initially we compute the similarity or weight to all other nodes $v_j, (j \neq i)$ based on the similarity of the presenting

feature vector of the group. But in this case, the attribute graph can be close to a complete graph and consequently it would increase the computational time. To avoid this problem, we propose to use a threshold for average degree of the attribute graph based on that of structure layer. Let average degree of the structure layer is avg_{st} , we only keep top avg_{st} edges that have the highest weight. Using avg_{st} has two advantages: (1) avg_{st} is determined value and (2) leveraging it allows us to maintain the balance between the size of structure layer and attribute layer. At the end, we get the attribute layer in the form of a graph $\mathbf{G}_{at} = (\mathbf{V}_{at}, \mathbf{E}_{at})$.

4.3 Connect Attribute and Structure Layer

Next, we complete the multi-layer network by adding the edges that connects structure layer and attribute. For any node v_i in the original network, suppose it corresponds to node v_i^{st} in the structure layer and node v_i^{at} in the attribute layer, we join v_i^{st} and v_i^{at} together by an edge which associated with four transition weights as follows:

$$w_i^{ss} = \ln(e + |\Gamma_i^{ss}|) \mid \Gamma_i^{ss} = \{(v_i^{st}, v_j^{st}), v_j^{st} \in \mathbf{V}_{st}\} \quad (1)$$

$$w_i^{sa} = w_i^{as} = \ln(e + |\Gamma_i^{at}|) \mid \Gamma_i^{at} = \{(v_i^{at}, v_j^{st}), v_j^{st} \in \mathbf{V}_{st}\} \quad (2)$$

$$w_i^{aa} = \ln(e + |\Gamma_i^{aa}|) \mid \Gamma_i^{aa} = \{(v_i^{at}, v_j^{at}), v_j^{at} \in \mathbf{V}_{at}\} \quad (3)$$

This weights are determined based on the number of direct neighbors in terms of topology and attribute. The intuition behind these weights is to guide the random walk generation process to the layer which have more information in each walk step. The detail of usage of these weights will be discussed later at Sect. 5.1.

We have generated a multi-layered graph $\mathbf{G}_{mul} = (\mathbf{V}_{mul}, \mathbf{E}_{mul})$ using network structure and attribute. Next, we leverage this graph to learn the latent representations for network nodes.

5 Structure Attribute Network Embedding

In this step, each network is embedded into a low-dimensional space by combining structure and attribute information where each node v_i is represented as a d -dimensional vector z_i . First we describe a biased random walks generator on the multi-layered graph, then we address the latent representation learning strategy.

5.1 Biased Random Walks

Algorithm 1 depicts how the random walks are generated. Given the multi-layer network from the previous step, we would want to produce a random walks set which plays the same role as corpus in language modelling. There are r random walks starting from each node in the multi-layered graph as root, each random walk has the length l .

Algorithm 1. Random walks construction

```

1: Input: The network  $\mathbf{G}_{mul} = (\mathbf{V}_{mul}, \mathbf{E}_{mul})$ ,
       $r$ : Number of random walk starting form each vertex,
       $l$ : Length of each random walk
2: Output: Random walks set  $C$ 
3:  $C = []$  ▷ Initialize the corpus
4: for  $n \in [1, r]$  do
5:   for  $v \in \mathbf{V}_{mul}$  do
6:      $W = [v], v_0 = v$  ▷ Initialize the walk
7:     for  $i \in [1, l]$  do
8:        $v_i = WALKSTEP(v_{i-1})$  ▷ Determine the next node in walk step
9:       Append  $v_i$  to  $W$ 
10:    Append  $W$  to  $C$ 
11: return  $C$ 

```

Lines 4–9 shows the basis of the generator. Line 4, line 5 specifies the loops with r and l . For each iteration, line 6 first initialize the walk. Then, given at a particular time step i of the random walk, we are at node v_i , with v_i either in the structure layer or in the content layer; the *WALKSTEP* function calculates to choose v_{i+1} to perform the next walk step in line 8. The function first calculate the probability of taking that step either into the structure layer or into the attribute layer, with the goal is to move to a layer which is more informative using the weights specified in Sect. 4.3. The transition probabilities are defined as:

$$p(v_{i+1} \in \mathbf{V}_{st} | v_i \in \mathbf{V}_{st}) = \frac{w_i^{ss}}{w_i^{ss} + w_i^{sa}} \tag{4}$$

$$p(v_{i+1} \in \mathbf{V}_{at} | v_i \in \mathbf{V}_{st}) = \frac{w_i^{sa}}{w_i^{ss} + w_i^{sa}} \tag{5}$$

$$p(v_{i+1} \in \mathbf{V}_{st} | v_i \in \mathbf{V}_{at}) = \frac{w_i^{as}}{w_i^{as} + w_i^{aa}} \tag{6}$$

$$p(v_{i+1} \in \mathbf{V}_{at} | v_i \in \mathbf{V}_{at}) = \frac{w_i^{aa}}{w_i^{as} + w_i^{aa}} \tag{7}$$

After selecting the layer, the *WALKSTEP* function pick randomly a node from neighbor set of the current node v_i in that layer to move to. The chosen node v_{i+1} then is appended to the walk in line 9. The loop continues until the length of the walk reaches l and the walk is added to the corpus in line 10. When all random walks are generated, the corpus is returned in line 11.

5.2 Latent Representation Learning

After having the random walk set generated, we leverage Skip-Gram with Negative Sampling (SGNS) technique [14] in language modelling to learn the embedding for network nodes. In language modelling, the main goal is to maximize the

likelihood of a sequence of words appearing in a document. In our work, generated biased random walks play the same role as short sentences in language modelling and the goal is to maximize the co-occurrence probability of all vertices v_1, \dots, v_n in the walk given the observing vertex v_0 . Mathematically, it can be written in the following form:

$$\underset{v_0}{\text{maximize}} \ Pr(v_1, \dots, v_n \mid v_0) \quad (8)$$

Skip-Gram model can be used to produce embedding for nodes given the random walk corpus C . For a vertex v_i , $\phi(v_i)$ represents the embedding of the node, we would like to maximize the probability of its neighbors in the walk. Given w being the window size of the node, the embeddings ϕ can be found by maximizing the following objective:

$$\underset{\phi}{\text{maximize}} \ \log(\Pr\{v_{i-w}, \dots, v_{i+w} \mid v_i\} \mid \phi(v_i)) \quad (9)$$

6 Node Alignment

To reconcile two latent representation spaces that are learned independently, we construct a pairwise mapping between two embedding spaces. The idea behind this technique is to facilitate reconciliation based on a part of groundtruth given as seed dictionary, called anchor nodes, denoted by A . Because we construct embedding for source and target network separately, these anchor nodes will be assigned to different embedding spaces, even if they are related to the same entities. As a result, these anchor nodes play an important role to align one embedding space to the other.

Let \mathbf{Z}_s and \mathbf{Z}_t denote the embedding matrices for source and target network, we learn the mapping function $\Theta : \mathbf{Z}_s \rightarrow \mathbf{Z}_t$ such that the embeddings of the anchor nodes A are closed in the common space. Our objective can be determined by the following loss function:

$$L = \sum_{v \in A} \|\Theta(z_v^s) - z_v^t\|_F \quad (10)$$

where $\|\cdot\|_F$ is the Frobenius norm.

The mapping function Θ can be either a linear function [2] or a multilayer perceptron [12]. Although the linear function $\Theta(\mathbf{Z}) = \mathbf{Z} \times \mathbf{W}$ is a simple model, we apply this model into our work because of the two reasons. First, according to [12], it is sufficient to obtain a good mapping. Second, with the linear mapping, the optimized solution can be found in an exact manner. It is worth noting that better mapping comes with the addition of orthogonality constraint to the mapping matrix \mathbf{W} , because orthogonal matrix helps to maintain distance between any two nodes across networks. The optimized orthogonal matrix \mathbf{W}^* can be obtained by using singular value decomposition. In more detail, let

$$\mathbf{U}\Sigma\mathbf{V}^T = \mathbf{E}_s\mathbf{E}_t^T \quad (11)$$

be the SVD decomposition of the matrix $\mathbf{E}_s \mathbf{E}_t^T$. Then the mapping matrix \mathbf{W} is calculated from \mathbf{U}, \mathbf{V}

$$\mathbf{W} = \mathbf{U} \mathbf{V}^T \quad (12)$$

After reconciling the two representation spaces, we calculate the alignment matrix \mathbf{S} by:

$$\mathbf{S}(u, v) = \mathbf{sim}(\phi(u), \phi(v)), u \in \mathbf{V}_s, v \in \mathbf{V}_t \quad (13)$$

with \mathbf{sim} is the similarity measure, for our work we choose cosine similarity. Then, we adopt a heuristic greedy algorithm [9] on the similarity matrix to obtain one-to-one alignments between the source and target networks.

7 Experiments

7.1 Experimental Setup

Datasets. In the experiments, we employ three real-world datasets to evaluate the performances of our model, including protein-protein interaction (ppi) [6], economic network (econ) [16] and brain network (bn) [1]. Starting from each real-world network $\mathbf{G}_s = (\mathbf{V}_s, \mathbf{E}_s, \mathbf{N}_s)$ with adjacency matrix \mathbf{D}_s as source network, we produce a permuted network $\mathbf{G}_p = (\mathbf{V}_p, \mathbf{E}_p, \mathbf{N}_p)$ with adjacency matrix \mathbf{D}_p by the following equation:

$$\mathbf{D}_p = \mathbf{P} \mathbf{D}_s \mathbf{P}^T \quad (14)$$

where \mathbf{P} is the permutation matrix, with $\mathbf{P}_{ij} = 1$ means that node i from the source network corresponds to node j in the target network, otherwise $\mathbf{P}_{ij} = 0$.

Baseline Methods. We compare against four well-known existing network alignment methods: (1) UniAlign, which applies alternating projected gradient descent on formulated bipartite network alignment model [10]; (2) FINAL, which defines a model with three criteria namely structure consistency, node feature consistency and edge feature consistency to tackle alignment problem on attributed networks [20]; (3) REGAL, which models alignment matrix using topology and nodes' feature similarity then employs low-rank matrix factorization approximation to speed up calculation [8]. (4) DeepLink, which generates the embeddings using skip-gram model then using auto-encoder and MLP to construct mapping function [21]. Besides, we use one variant of our method that using only structure layer to verify the effectiveness of the construction of the multi-layer graph. We name it as RAN_{so} (RAN with structure layer only).

Settings. Due to the randomness, we run 50 times for each dataset to compute the average results. For our algorithm, we use following hyperparameters: number of random walks = 100, walk lengths = 5, window size = 2. For other algorithms, we try our best to tune the parameters to have the best experiment performance. We use alignment accuracy as the evaluation metric to measure the performance. All the experiments are conducted on an AMD Ryzen ThreadRipper 3.8 GHz system with 64 GB of main memory and four GTX Titan X graphic cards.

7.2 Alignment Performance Analysis

Robustness to Structural Noise. Network alignment methods exploit the topology consistency in its model, which assumes that the two node neighbors tends to maintain their relationship across the networks. Therefore, it would be important to learn about the effect of structural noise to these state-of-the-art alignment algorithms. To stimulate structural noise, we permute the original graph then remove edge from the original network with probability p_e ranging from 0 to 0.2 as described in 7.1

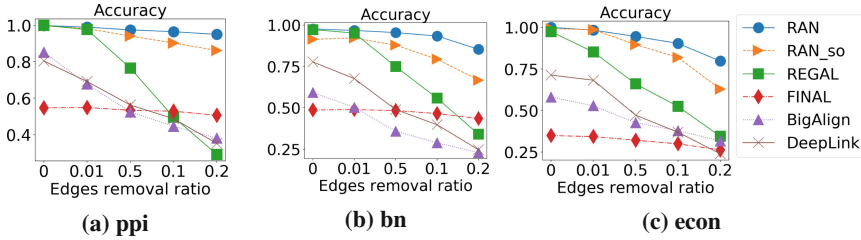


Fig. 2. Robustness of algorithms to structural noise

Figure 2 illustrates the accuracy results on the datasets. In general, all algorithms suffer accuracy drop when the noise level increased. RAN outperforms all four existing methods in all scenarios. It achieves 20–40% accuracy improvement over the state-of-the-art method REGAL when the noise level is at around from 0.05 to 0.1. This is because while both algorithm use network structure and attribute information, REGAL adopt a strict assumption on topology consistency that two nodes are similar when their neighbor’s degree are the same, which make its model susceptible to considerable level of structure noise. Furthermore, RAN is more robust to structural noise as the accuracy gap between it and other methods becomes larger when noise grows. The performance of FINAL also witnesses a slight decrease but remains stable at average level, around 50% for ppi and bn dataset and 40% for econ dataset. By contrast, the accuracy of REGAL, BigAlign and DeepLink drop sharply when the level of noise goes high. Last but not least, the alignment accuracy of RAN is 10–30% better than its variant RAN_so, which verifies the importance of the construction of multi-layer graph.

Robustness to Attribute Noise. In this experiment, we study the effect of attribute noise to the performance of RAN and other baseline methods. Figure 3 shows the effect of attribute noise on alignment algorithms with the attribute changing p_a probability ranging from 0 to 0.5 while the structural noise p_s fixing at 0.05. We only consider RAN, REGAL, FINAL and BigAlign because the others do not utilize attribute information.

It can be seen from the diagram that REGAL and FINAL performance deteriorate with similar rate when level of attribute noise rises, while BigAlign’s

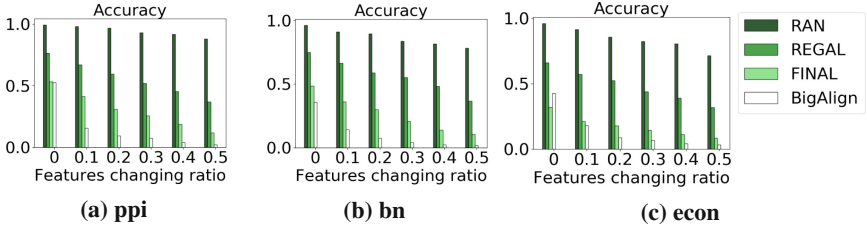


Fig. 3. Robustness of algorithms to attribute noise

accuracy declines with slightly faster pace. RAN performs the best not only on the accuracy but also on the stability when the level of attribute noise increased for all three datasets. On the other hand, performance of FINAL and BigAlign decrease significantly when attribute noise arises, drop below 10% accuracy at the noise level of 0.5 for all datasets. This is expected due to FINAL and BigAlign heavy reliance on attribute consistency.

Robustness to Graph Size Imbalance. This experiment investigates the effect of the difference between the size of two network to the accuracy of all models. Figure 4 shows the experiment result with the removal ratio of nodes p_n increases from 0 to 0.5.

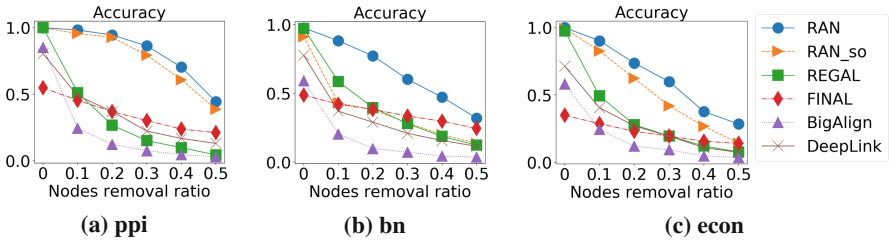


Fig. 4. Robustness of algorithms to graph size imbalance

It can be observed from the diagram that representation based method such as RAN, RAN_{so} and REGAL are sensitive to graph size imbalance factor. On the other hand, FINAL, a matrix-factorization based method, is more stable than the other when the level of imbalance increased. This is because size imbalance between source and network make the corresponding representation vector space harder to reconcile. It is worth notice that RAN is not the winner on the stability of performance when the number of removal nodes increased but it performs the best in accuracy on all datasets.

8 Related Works

Network alignment problem has received a great deal of research interests in the recent years. The problem appears in various contexts, ranging from data mining [3], database schema matching [13], computer vision [19], security [4] to bioinformatics [17].

Many approaches use matrix factorization to compute the alignment matrix directly, which is natural but effective way to tackle the problem. The classic and well-known IsoRank algorithm [17], inspired by PageRank [18], propagates the pairwise node similarity along the network with the assumption that two corresponding nodes in two networks connect to similar characteristic neighbors. NetAlign [3] models the alignment problem as an integer quadratic programming problem and adopts a belief propagation heuristic to solve. UniAlign [10] then applies alternating projected gradient descent on formulated bipartite network alignment model. FINAL [20] defines a model with structure, node feature and edge feature consistency to tackle alignment problem. REGAL [8] employs low-rank matrix factorization approximation to speed up calculation.

While matrix factorization based methods prove their efficiency in many scenario, they struggle to deal with large-scale networks due to the sparsity and massive size of their adjacency matrix. With the appearance of network embedding techniques [5, 6, 15]; alignment techniques leverage their scalability to deal with large-size network. PALE [12] learns nodes embedding by maximizing the co-occurrence likelihood of edge's vertices then applies linear or multilayer perceptron (MLP) as mapping function. IONE [11] uses the same mapping function as PALE but its embedding process is more complicated as it takes into account second-order node similarity. DeepLink [21] employs unbiased random walk to generate embeddings using skip-gram then using auto-encoder and MLP to construct mapping function. While these methods put in solid performance in some large datasets, they rely only topology information and therefore remain vulnerable to structure noise, which is very common in real-world networks. In this paper, in our end-to-end setting, we integrate both structure and attribute information to mitigate this problem and enhance the alignment result.

9 Conclusion

This paper proposed a representation learning based technique to align two attributed networks. The main novelty of our work is to transform the given networks into a multi-layer network to integrate both topology and attribute information before mapping network nodes to latent representation vector space. Then, the alignment matrix is retrieved by reconciling the two embedding spaces through a mapping function and eventually matching heuristic is applied on the matrix to retrieve the corresponding node pairs. Experiments on different benchmark datasets, different settings and numerous alignment methods verify the efficiency of our method on real-world large scale networks.

References

1. Amunts, K., et al.: BigBrain: an ultrahigh-resolution 3D human brain model. *Science* **340**, 1472–1475 (2013)
2. Artetxe, M., Labaka, G., Agirre, E.: Learning bilingual word embeddings with (almost) no bilingual data. In: *ACL*, pp. 451–462 (2017)
3. Bayati, M., Gerritsen, M., Gleich, D.F., Saberi, A., Wang, Y.: Algorithms for large, sparse network alignment problems. In: *ICDM*, pp. 705–710. IEEE (2009)
4. Bayati, M., Gleich, D.F., Saberi, A., Wang, Y.: Message-passing algorithms for sparse network alignment. *ACM Trans. Knowl. Discov. Data (TKDD)* **7**(1), 3 (2013)
5. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: *KDD*, pp. 855–864. ACM (2016)
6. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: *NIPS*, pp. 1024–1034 (2017)
7. Hashemifar, S., Xu, J.: HubAlign: an accurate and efficient method for global alignment of protein-protein interaction networks. *Bioinformatics* **30**(17), i438–i444 (2014)
8. Heimann, M., Shen, H., Safavi, T., Koutra, D.: REGAL: representation learning-based graph alignment. In: *CIKM*, pp. 117–126 (2018)
9. Kollias, G., Mohammadi, S., Grama, A.: Network similarity decomposition (NSD): a fast and scalable approach to network alignment. *TKDE* **24**(12), 2232–2243 (2012)
10. Koutra, D., Tong, H., Lubensky, D.: BIG-ALIGN: fast bipartite graph alignment. In: *ICDM*, pp. 389–398 (2013)
11. Liu, L., Cheung, W.K., Li, X., Liao, L.: Aligning users across social networks using network embedding. In: *IJCAI*, pp. 1774–1780 (2016)
12. Man, T., Shen, H., Liu, S., Jin, X., Cheng, X.: Predict anchor links across social networks via an embedding approach. In: *IJCAI*, vol. 16, pp. 1823–1829 (2016)
13. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In: *ICDE*, pp. 117–128 (2002)
14. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: *NIPS*, pp. 3111–3119 (2013)
15. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: *KDD* (2014)
16. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: *AAAI* (2015). <http://networkrepository.com>
17. Singh, R., Xu, J., Berger, B.: Global alignment of multiple protein interaction networks with application to functional orthology detection. In: *Proceedings of the National Academy of Sciences*, pp. 12763–12768 (2008)
18. Xing, W., Ghorbani, A.: Weighted PageRank algorithm. In: *CNSR*, pp. 305–314. IEEE (2004)
19. Yang, H., Song, D., Liao, L.: Image captioning with relational knowledge. In: Geng, X., Kang, B.-H. (eds.) *PRICAI 2018*. LNCS (LNAI), vol. 11013, pp. 378–386. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-97310-4_43
20. Zhang, S., Tong, H.: Final: fast attributed network alignment. In: *KDD*, pp. 1345–1354 (2016)
21. Zhou, F., Liu, L., Zhang, K., Trajcevski, G., Wu, J., Zhong, T.: DeepLink: a deep learning approach for user identity linkage. In: *INFOCOM*, pp. 1313–1321 (2018)