**­HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**GRADUATE THESIS**

**DEVELOPMENT OF GRAPH ALIGNMENT ARCHITECTURE FOR ATTRIBUTED GRAPHS**

**TỐNG VĂN VINH**

vinh.tv154381@sis.hust.edu.vn

**Information Technology and Communication**

**Major: Information Technology**

|  |  |
| --- | --- |
| **Supervisor:** | Assoc. Prof: Huỳnh Quyết Thắng \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **Institute:** | School of Information and Communication Technology |
| **HANOI, 6/2020** | |

**GRADUATION TOPIC**

Network alignment on attributed graphs

**PLEDGE OF COMMITMENT**

Supervisor

Signature

*1.* *Student Information*

Student full name: Tong Van Vinh

Contact number: 0354095052 Email: [vinh.tv154381@sis.hust.edu.vn](mailto:vinh.tv154381@sis.hust.edu.vn)

Class: KSTN-CNTT K60 Study mode: Full-time Engineer

The thesis was performed in Hanoi

Thesis period: From 14/02/2020 to 26/06/2020

*2. The purpose of the thesis*

The thesis gives a deep insight into the graph alignment problem. From definition to several current challenges in solving this task. Moreover, some research approaches have been investigated in detail. On top of that, A novel framework named *GAlign* has been proposed as an unsupervised network alignment architecture for attributed network. The thesis shows that *GAlign* returns state-of-the-art performance.

*3. This thesis’s tasks in detail*

- Do a research about the network alignment problem on graphs.

- Give the definition of network alignment problem on attributed networks.

- Investigate current approaches tackling this problem.

- Propose *GAlign*, a graph alignment architecture, to solve this problem.

- Construct various experiments on both synthetic datasets as well as real-world datasets and give insight evaluations.

- Applythe proposed model to solve a network alignment – related problem.

- Give the summarization of the thesis and explore a new direction to improve the model.

*4. Student’s commitment*

I – Tong Van Vinh – pledge that the Graduation Thesis is my research under the guidance of Assoc. Prof. Huynh Quyet Thang. The results stated in the thesis are honest, my results, not reproduced by any other works. All citations in the thesis are clearly and fully documented in the REFERENCES section. I take full responsibility for even one copy that violates the school rules.

Hanoi, June 26, 2020

Author of this Thesis

Tong Van Vinh

*5. Confirmation of supervisor about how well the thesis have been done and allow the author to present the work.*

*………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………*

Hanoi, June 26, 2020

Supervisor

Assoc. Prof. Huynh Quyet Thang

**Acknowledgements**

I would like to extend thanks to the many people, who so generously contributed to the work presented in this graduate thesis.

Special mention goes to my supervisor, Assoc. Prof Huynh Quyet Thang and my mentor, Huynh Thanh Trung. This semester is an amazing experience and I really appreciate my two supervisors for their wholehearted support and many wonderful opportunities giving to me.

**Abstract**

Networks are natural but powerful structures that capture relationships between different entities in many domains, such as social networks, citation networks, bioinformatic networks. In many applications that involves multiple networks analysis, network alignment, the task of recognizing node correspondence across different networks, plays an important role. Given the important applications of network alignment, there is a rich body of researches that aim to tackle this problem. However, many research challenges still exist that need to be resolved, such as enhancing the accuracy, handling the information explosion in complex system, improving the existing algorithms in term of time and memory efficiency. Successfully conquering these challenges will make great impact network science and applications varying in different domains in our daily life.

With such motivation, we address the two crucial challenges, namely improving accuracy of network alignment and designing adaptive alignment network on attributed networks. We have followed this research topic for two years and have achieved several fruitful results, including 1 A\*-ranked and 1 B-ranked conference, 1 Q1 journal published and 1 Q1 journal being revised. The contributions from these publications will be presented in this graduate thesis.

Student

Signature

Sinh viên thực hiện

Ký và ghi rõ họ tên

**CONTENTS**

[CHAPTER 1. INTRODUCTION 1](#_Toc43988779)

[1.1 Network Alignment definition 1](#_Toc43988780)

[1.2 Network Alignment Applications 2](#_Toc43988781)

[1.2.1 Social network analysis 3](#_Toc43988782)

[1.2.2 Bioinformatic networks analysis 4](#_Toc43988783)

[1.2.3 Knowledge Graph Alignment 4](#_Toc43988784)

[1.3 Research Goals 5](#_Toc43988785)

[1.4 Contributions 6](#_Toc43988786)

[1.5 Report outline 8](#_Toc43988787)

[CHAPTER 2. BACKGROUND 9](#_Toc43988788)

[2.1 Spectral methods 9](#_Toc43988789)

[2.1.1 Structural consistency based techniques 9](#_Toc43988790)

[2.1.2 Attribute consistency based techniques 10](#_Toc43988791)

[2.1.3 Embedding driven techniques 12](#_Toc43988792)

[2.2 Representation learning methods 14](#_Toc43988793)

[2.2.1 First-order proximity based techniques 14](#_Toc43988794)

[2.2.2 Second-order proximity based techniques 15](#_Toc43988795)

[2.2.3 Hybrid techniques 17](#_Toc43988796)

[CHAPTER 3. PROPOSED TECHNIQUE 20](#_Toc43988797)

[3.1 Model and Problem Statement 20](#_Toc43988798)

[3.1.1 Attributed network 20](#_Toc43988799)

[3.1.2 Network alignment 21](#_Toc43988800)

[3.1.3 Alignment constraints 21](#_Toc43988801)

[3.2 Approach overview 22](#_Toc43988802)

[3.2.1 Motivation 22](#_Toc43988803)

[3.2.2 Solution Sketch 23](#_Toc43988804)

[3.2.3 End-to-end network alignment framework 24](#_Toc43988805)

[3.3 GCN for consistent network alignment 24](#_Toc43988806)

[3.3.1 GCN model 24](#_Toc43988807)

[3.3.2 Permutation Immunity 25](#_Toc43988808)

[3.3.3 From consistency to embedding 26](#_Toc43988809)

[3.4 Multi-order embedding 26](#_Toc43988810)

[3.4.1 GCN-based embedding model 26](#_Toc43988811)

[3.4.2 Consistency loss 27](#_Toc43988812)

[3.4.3 Put It Altogether 28](#_Toc43988813)

[3.5 Alignment Computation 28](#_Toc43988814)

[3.5.1 Alignment Instantiation 28](#_Toc43988815)

[3.5.2 Alignment refinement 29](#_Toc43988816)

[3.5.3 Complexity Analysis 32](#_Toc43988817)

[CHAPTER 4. EXPERIMENT EVALUATION 33](#_Toc43988818)

[4.1 Experiment setup 33](#_Toc43988819)

[4.2 End-to-end comparison 35](#_Toc43988820)

[4.3 Ablation test 37](#_Toc43988821)

[4.4 Adaptivity to adversarial conditions 37](#_Toc43988822)

[4.5 Hyperparameter sensitivity 40](#_Toc43988823)

[4.6 Qualitative study 41](#_Toc43988824)

[CHAPTER 5. KNOWLEDGE GRAPH ALIGNMENT 43](#_Toc43988825)

[5.1 Knowledge graph representations 43](#_Toc43988826)

[5.2 Knowledge Graph Alignment with GAlign 44](#_Toc43988827)

[5.3 Experiment Evaluation 44](#_Toc43988828)

[CHAPTER 6. CONCLUSION 47](#_Toc43988829)

[6.1 Conclusion 47](#_Toc43988830)

[6.2 Future Direction 47](#_Toc43988831)

**LIST OF FIGURES**

[Figure 1: Example of Network Alignment 2](#_Toc43985017)

[Figure 2: Overview of FINAL algorithm 12](#_Toc43985018)

[Figure 3: Overview of REGAL algorithm 13](#_Toc43985019)

[Figure 4: Overview of PALE algorithm 15](#_Toc43985020)

[Figure 5: Overview of DeepLink algorithm 16](#_Toc43985021)

[Figure 6: Structural and attribute consistency example 21](#_Toc43985022)

[Figure 7: Overview of GAlign framework 24](#_Toc43985023)

[Figure 8: Robustness against structural noises 38](#_Toc43985024)

[Figure 9: Robustness against attribute noises 39](#_Toc43985025)

[Figure 10: Robustness against isomorphic level 39](#_Toc43985026)

[Figure 11: Effects of #GCN-layers against Success@1 (w.r.t embeddings used to compute alignment matrix) 40](#_Toc43985027)

[Figure 12: Embedding dimension 41](#_Toc43985028)

[Figure 13: Qualitative study 42](#_Toc43985029)

[Figure 14: Example of knowledge graph entity alignment 43](#_Toc43985030)

**LIST OF TABLES**

[Table 1: Notation Summary 20](#_Toc43985031)

[Table 2: Statistics of real-world networks 33](#_Toc43985032)

[Table 3: Network alignment comparison on real-world datasets 36](#_Toc43985033)

[Table 4: Ablation test 37](#_Toc43985034)

[Table 5: Layer Weights 40](#_Toc43985035)

[Table 6: Statistics of real-world datasets 45](#_Toc43985036)

[Table 7: End-to-end effectiveness 46](#_Toc43985037)

# INTRODUCTION

This report covers different aspects of network alignment problem, the task of identifying node correspondences across different graphs. In this chapter, we first try to give the easy to understand definition of the term “network alignment” and introduce several of its applications to demonstrate the important role of the problem. The remain of this chapter introduces some of network alignment applications, it also addresses several challenges from what we draw our research goal.

## Network Alignment definition

Networks a.k.a. graphs, are ubiquitous data structure that are used to represent complex social, technological, and biological systems appearing in every aspect of our daily life, ranging from Internet, social network, road network, trade networks to interacting genes and proteins networks. Thanks to network, we can not only capture the information of entities in real-world by using *nodes* but also model the complex relationships between them by using *edges*. *Network Science,* a research field aiming to study several computational, algorithmic, and modeling challenges, has been an active area of research for many years.

As a research branch of *Network Science,* **Network Alignment**, the task of identifying nodes that belong to the same entity from different networks, is one of fundamental problem appearing in a significant number of data analysis applications. 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 [1]. In computer vision, network alignment helps to match images without human supervision [2]. In bioinformatics, analysis of protein-protein interactions networks among species makes remarkable improvement for gene prioritization [3]. Further details of network alignment applications will be discussed in section 1.2.

Despite its ubiquity, network alignment problem is a task of challenge. The simplest method which uses only names or labels of nodes to match entities across different networks is unreliable because those labels are generally unavailable or incomparable directly on different graphs. Instead, current robust architectures often exploit topology structure of graphs as well as pre-defined node attributes to tackle this challenge. Existing works observed two types of constraints that should be held to achieve high performance that are *structural consistency* and *attribute consistency* principles. The structural consistency is often referred as the *homophily rule:* If two nodes have close relation in the topology in one network, their anchor nodes in the other network shall maintain this relation [4]. For example, if two persons are friends in one social network, they are likely to connect in another social network. On the other hand, the attribute consistency states that corresponding nodes shall share the same attribute values [5]. For example, the profile information (e.g. age, email address) of a single person in different social networks shall be similar. Figure 1 demonstrates a toy example of network alignment on two small networks as well as structural and attribute consistency principle. The source graph and the target graph each has five nodes and six edges. After the alignment process, node 1, 2, 3, 4, 5 of are matched with node 1`, 2`, 3`, 4`, 5` of respectively. The structural consistency is respected, for instance the neighborhood of node 1 with node 2, 3, 4, 5 are preserved as node 1` also having the connection with node 2`, 3`, 4`, 5`. The attribute consistency principle is also fulfilled as the matching nodes sharing the same attribute. However, these principles sometimes are violated in real-world datasets, which often is referred as *structural noise* and *attribute noise*. Therefore, a solid network alignment technique must handle and be robust to these noises.

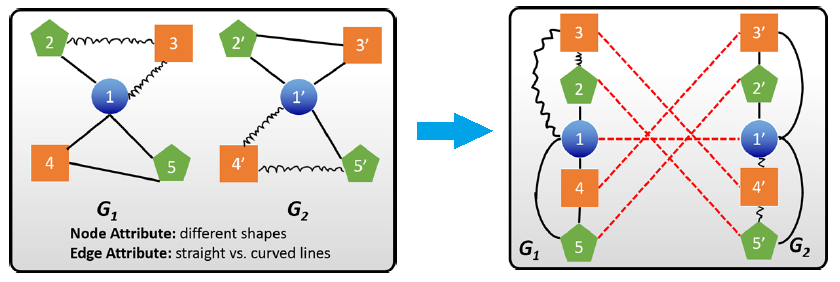


Figure 1: Example of Network Alignment

The complexity of real-world systems grows overtime, which means the size of graphs used to represent these systems becomes really large. For instance, social networks such as Facebook or Twitter contain billions of nodes and edges while they have to process user requests in real time. Therefore, there is a crucial need for alignment techniques to handle large-scale networks with low time and memory complexity, which traditional algorithms often struggle with. Moreover, our world is changing over time, which means our systems are not only complex by its huge volumes of entities and relationship but also by its dynamic nature. Understanding of dynamic properties of large complex networks is an arising challenge for network science in general and network alignment in particular. To the best of our knowledge, there are little techniques considered this aspect of network.

Given these challenges, in this research semester, along with results we have achieved so far, we investigate network alignment problem on general as well as heterogeneous graphs such as knowledge graphs. We believe the improvement and contribution to this problem will make impact on various domains.

## Network Alignment Applications

In this section, we consider three important applications domain of network alignment, namely social network analysis, bioinformatic network analysis and knowledge graph analysis.

### Social network analysis

Social networks are ubiquitous and there is a significant number of users who have accounts different social networks [6]. Given this phenomenon, there is a need to align users from different social networks [7], which is a crucial step in cross-network recommendation [8] or link prediction [9], [10], [11], [12], [13]. For instance, if two users are friends on one social network, it is more likely that they are also friends on the other network. Another example is to recommend a product to a user in a sparse social network based on the information obtained on the user on another social network [14].

The rapid increase in the number of users with accounts on multiple social networks results in new chances and challenges. First, users with accounts on multiple social sites of different purposes give potential to better understanding of user behaviors and thus provide better recommendations or services. To achieve this, we have to recognize the identical user accounts across the networks then collect and analyze their data on these social media sites together to have a more comprehensive view about the user. The network alignment problem is the fundamental step in such process. Second, matching the accounts from same user allows us to integrate patterns among online social network sites and overcome hard problems such as data sparsity and cold start in many predictive tasks. For example, a brand new social media service may lack historical data for recommendations to users. Thus, if we can identify these users on other mature sites, we can transfer knowledge from these sites to the new site and mitigate the problems. Finally, recognizing users with accounts on multiple social media sites can also help to analyze user migration patterns and guide web developments.

In terms of downstream functions in social network site that network alignment is a helpful solution, we would like to address a few:

* **Friend Recommendation.** Regarding to homophily principle, two users have connection on one network will have high chance to maintain their neighborhood relationship on others network. Therefore, if the alignment can be performed on two networks with high accuracy, the friendship of user in one site can be used to enhance the performance of friend recommendations on the other site and vice-versa.
* **Information diffusion.** Information such as rumors can flow within and across different social networks. Therefore, it is interesting to investigate whether information diffuses more within one network or across networks. These mechanisms will make great impact on the performance of various downstream tasks such as information cascade tree construction [15].
* **Analyzing Network Dynamics.** Dynamics of single-site social networks are well-studied in the literature, such as power-law degree distribution, small average path length, high clusterability. The natural but effective idea is to generalize these network properties to multiple networks. Thus, it is interesting to determine how close the dynamics of single networks (e.g. degree distribution) are to that of multi-networks.

### Bioinformatic networks analysis

Proteins are large biomolecules that perform vital functions in living cells. Proteins rarely operate alone, they often interact with each other, which enables their diverse functionality [16], such as forming signaling networks and metabolic pathways, regulating enzymatic activities. The interactions between proteins often is known as protein-protein interaction (PPI).

PPI networks are a valuable source of information for understanding the evolution of protein interactions and system-level cellular processes. The analysis of common structure shared between PPI networks provides insight into species evolution and correlation, such as pathways across multiple species [17], [16]. Network alignment algorithms play an important role in such applications.

The applications using network alignment to compare PPI networks have achieved some remarkable results, especially in discovering common high-sequence interaction patterns between different species. A typical success example is the detection of similar protein interactions in protein orthologs between yeast and worms [18].These functional interactions are only conserved among species sharing the evolutive connections, thus network alignment can also be used to verify and increase the precision of observed interaction in biology databases [19]. PPI-network alignment also has applications in many other domains such as protein functions prediction, protein orthology detection, protein-protein interaction prediction and drug design [20].

### Knowledge Graph Alignment

Knowledge graphs (KGs) are used to represent real-world entities and the relationships between them and between their own attributes [21], [22]. Various applications have exploited such generality of KGs to model their data such as recommendation systems [23], text classification [24], question-answering [25], semantic search [26] and knowledge reasoning [25]. To make these applications more ubiquitous, many multilingual knowledge bases come to existence, including DBpedia, YAGO, and BabelNet, in which a KG is built for each human language [27]. However, these monolingual KGs are often disconnected due to semantic heterogeneity and the huge labor effort to align them. Entity alignment, the task of identifying corresponding entities between monolingual KGs, is the foundation to integrate multiple KGs [28]. A wide range of knowledge- and data-base communities can use the alignment result for further data enrichment steps, such as translating knowledge, repairing inconsistencies, filling knowledge gaps, and building cross-lingual applications [29].

The problem of entity alignment for cross-lingual KGs has been studied intensively with the emergence of graph embedding techniques. Given two monolingual KGs, these techniques first learn low-dimensional vectors representing the entities of each KG, then the corresponding entities are discovered based on their vector similarity. The first generation of this paradigm, including MTransE [28], JAPE [30], ITransE [31], and BootEA [32], learn the embeddings with the assumption that if two entities have a relationship, the distance between their respective embeddings is equal to the relation embedding. Avoiding this strict assumption [33], the second generation of embedding techniques such as GCN-Align [27], RDGCN [33], MUGNN [34], KG-matching [35], and NAEA [36] employ graph neural networks, which encode structural relationship via neighborhood information.

## Research Goals

As mention in section 1.1, network alignment is notoriously a hard problem due to its NP-hard nature and the fast expansion of networks in complex applications nowadays. In this section, we discuss briefly about some existing challenges of network alignment on graphs that are addressed in the scope of our work.

Nowadays, we are witnessing rapid increase in the amount of published information, which often refer to the term *information explosion*. Real-world systems also are complex than the general ones that are often studied. For example, social networks such as Facebook and Instagram can contain several node types namely, user-nodes, fanpage-node, group-node to name but a few. Therefore, to model those complex systems, we have to solve the problem on heterogenous graphs or graphs with addition nodes and edges attribute. Hence, one of the main challenges we want to address in our work is to design network aligner model for attributed graphs.

To achieve this task, we have to answer several research questions. The first question is how to guarantee our model can take advantage of both topology information as well as node attribute information. Traditional network alignment techniques using representation learning, which allow to present network nodes in low-dimension vector space, have proved their efficiency in non-attributed graphs, but they struggle to make use of node attribute information in order to enhance their alignment performance. Although there are several matrix factorization techniques which have proved their efficiency in attributed graphs, but these graphs are small, and they struggle to scale up because of their high computational complexity due to the sparsity and enormity of adjacency matrix. Recently, the rise of interconnected multi-core processors/graphic cards and distributed systems has strengthened the computational power for the system and has brought a significant number of opportunities to process very large networks.

The second question we have to answer is how to make the techniques robust to noises, such as structural and attribute noises. The network alignment techniques often base on topology and attribute consistency, which are introduced in section 1.1. However, with the development of information networks, the pattern of user behaviors varies so that these principles might not be respected in many cases in real-world datasets. For example, one user might be very active on a social network such as Facebook but stay quieter in another network. Therefore, the designed network aligner model has to cope with and mitigate this problem.

The third question is how to exploit other type of information in common between networks. As we discuss above, relying on solely topology and attribute consistency make the model susceptible to noises. Therefore, it would be great to consider other kind of information. One great example is to consider the hierarchical structure of the networks. In real-world networks, entities often gather into communities, such as people in one group in Facebook, proteins in local molecular structure in enzyme networks [37]. Discovering and integrating such great information will make great boost to the performance of network aligner.

## Contributions

Although a significant number of studies have been done to tackle the network alignment problem, there are still a lot of challenges related to efficiency, the richness of incorporated information, and the strictness of alignment constrains that need to be solved. We can formulate the network alignment problem as a maximum bipartite matching problem. But this had been demonstrated to be a NP-hard problem due to the complex nature of graphs [43]. A lot of works based on matrix factorization, e.g., IsoRank [44], NetAlign [43], UniAlign [45], FINAL [1], and REGAL [5] fail to deal with large scale graphs due to high computation complexity which grows dramatically with the network size (e.g., cubic growths for FINAL [1]).

Moreover, networks often comprise heterogeneous information including network structure and node features. While different types of information may be useful for network alignment, the lack of common modality imposes challenges [46]. Information integration may be guided by path-based constrains and proximity rules [47]. Yet, such approaches are domain-specific and require manual user efforts.

Concerning the constraints used for alignment, most techniques employ some strict notion of structural consistency, e.g., the relation between pairs of nodes shall always maintained across two aligned networks. However, such strict constraints are sensitive to network perturbations. For example, a person may have more connections in one social network (e.g., Facebook) than in others (e.g., LinkedIn, YouTube). Moreover, node pairings can also be one-to-many, especially for networks with different sizes. In that case, alignment heuristics that employ strict notions of structural consistency, such as a one-to-one constraint [48], are no longer applicable.

To address the outlined challenges, we propose an embedding-based network alignment model. In essence, our idea is to embed nodes of two networks based on the similarity of their node embeddings. We realize this idea through the design of a graph convolutional network (GCN) to learn embeddings. More specifically, we summarize our contribution as follows:

* We propose **GAlign**, a model based on graph convolutional network to align two or more graphs. Our model is unsupervised, which means no prior information about the relation of the networks (aka anchor links). With the advantage of graph neural network, our model can combine both structural information as well as prior node attribute to create low-dimensional representation of nodes.
* We demonstrate that the multi-order nature of GCN can be useful for network alignment task. Based on that, we propose a specific way to exploiting that interesting characteristic in network alignment problem. We show theoretically that the consistency constrains used separately by existing techniques can be unified in the same GCN model. This enables us to incorporate these constrains directly during node embedding, thereby supporting expressive constraints beyond strict structural consistency.
* Using this theory, we design a specific GCN model for network alignment. Traditional representation learning techniques often separate the alignment process into two main steps. The first step is to learn node representation and the second step is to map embedding spaces into one vector space. Our model learns embeddings jointly for the networks so that reconciliation of their embedding spaces is no longer needed.
* We evaluate our proposal in extensive experiments on real and synthetic networks. Moreover, we highlight an application of our model by solving a problem named “Knowledge graph alignment”. Our framework not only achieves superior performance compared to baseline techniques but also exhibits robustness to noise and network size imbalance.

In the scope of network alignment, we have fortunately achieved several results. Three publications have been accepted namely:

* *A comparative study on network alignment techniques* [44]
* *Network Alignment by Representation on Structure and Attribute* [45]
* *Adaptive Network Alignment with Unsupervised and Multi-order Convolutional Networks* [46]

Moreover, we have submitted an additional paper, which are in the review process:

* *Aligning isomorphic networks with self-supervised anchor links*

The content of this thesis mainly come from the works of our two published papers [44], [45]. The model *GAlign* is the main work of the paper *A comparative study on network alignment techniques.* And our investigation about several state-of-the-art models in the next section of this thesis have been done in the paper *Adaptive Network Alignment with Unsupervised and Multi-order Convolutional Networks.* We hope that our contributions will make this thesis valuable for not only graph-related field research communities but also *R&D* comunities in industry.

## Report outline

Further in this report, we first provide a literature survey in CHAPTER 2 that introduces the state-of-the-art alignment methods under two categories, namely spectral methods and representation learning methods. Then, we discuss about some current research challenges that the current works have not succeeded to address. In CHAPTER 3, we give a detail description of our proposed model to solve the problem. CHAPTER 4 contains our various experiments conducting on both real-datasets and synthetic-dataset. To emphasize the power of our model, we use *GAlign* to perform different alignment tasks on a real-world problem which can be modeled as graph alignment task, in which the detail can be found in CHAPTER 5. Finally, the conclusion of our work as well as the future potential directions will be addressed in CHAPTER 6.

# BACKGROUND

Due to the important applications of network alignment, there is a rich body of researches that aim to tackle this problem [38]. These techniques can be classified into two categories including *spectral methods* and *network representation learning methods*. Spectral methods aim to align two networks based on manipulation of the adjacency matrices. As each network is captured by its adjacency matrix, spectral methods can be considered as a direct way to perform alignment between two networks [39]. On the other hand, network representation learning methods require an intermediate step in which the nodes in the networks are represented as embeddings [40]. These embeddings allow to capture network structure information [41] and possible node features if available [42]. Given the embeddings of the two networks, a mapping step is performed to identify node correspondences.

## Spectral methods

Spectral methods using matrix factorization with the aim to compute the alignment matrix S directly. In many cases, these methods take into account the adjacency matrices of the two networks and denoted as and .

### Structural consistency based techniques

Traditional techniques are often based solely on structural consistency to calculate alignment result. We here present IsoRank, a well-known and typical algorithm in the literature, to illustrate how structural consistency based techniques work

**IsoRank.**The general idea of IsoRank is that two nodes from two networks are similar if their neighborhoods are similar as well. As a result, the similarity between two nodes depends on the similarity between their neighbors. Based on this observation, IsoRank defines the similarity between node and as follows:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where denotes the set of neighbors of .

The above formula can be rewritten in its matrix form as:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

In order to find , IsoRank follows an iterative approach where the alignment matrix is updated iteratively as follows:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where is the alignment matrix at iteration .

### Attribute consistency based techniques

Besides structural consistency, node and edge attribute consistency are also an important source of information that spectral network alignment techniques often exploit. In this section, we present BigAlign, a technique considers only attribute consistency; and FINAL, a technique makes use of both structural and attribute similarity among the networks.

**BigAlign.** BigAlign differs from the above methods significantly as it aims to solve the network alignment problem by converting to bipartite graphs. Given the specific structure of bipartite graphs, BigAlign hopes to achieve better alignment quality. In addition, instead of computing directly the alignment matrix , BigAlign takes a different perspective. BigAlign considers the alignment problem to be the reordering of the rows and columns of the adjacency matrix such that the reordered is similar to . Formally, the problem BigAlign wants to solve is to find a permutation matrix such that the following function is minimized:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

In this case, is used to reorder the rows of while is used to reorder the columns of it.

However, as this problem is hard to solve given its permutation nature, BigAlign proposes to convert the graphs into bipartite graphs and then optimize the following loss function:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | | Eq. . |
|  | | |  | Eq. . |

where is the trace of a matrix and is the sparsity constraint which enforces the matrix to be sparse. Note that the matrices in the above equation are not permutation matrices as in Eq. 2.5. As a result, these matrices are very large given the size of the graphs which makes the sparsity constraint important as we want large matrices to be sparse. As the graphs are bipartite, can be considered as how to reorder one side of the bipartite graph to match one side of while is responsible for matching the other sides of and .

The above loss function can be optimized using alternating projected gradient descent (APGD) using the following update steps:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |
|  |  | Eq. . |

where is the matrix at the -th iteration and are the step sizes of the APGD algorithm.

Then, the problem of unipartite graph alignment can be solved by carefully converting unipartite graphs into bipartite graphs. BigAlign proposes to construct features of unipartite graphs such as node degrees, clustering coefficients. As a result, both , can be modelled as bipartite graphs with the same set of features. This means there is no need to align the feature side of the bipartite graphs , as they are already the same i.e. is an identity matrix. As a result, we only need to find which can be found by taking the derivative of Eq. 2.5:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where is the pseudo-inverse of a matrix.

**FINAL.** FINAL defines three criteria to align two networks which are structure similarity, node feature similarity and edge feature similarity. Two pairs of nodes from two graphs are similar structurally if are close in then are also close in . On the other hand, edge feature similarity states that feature of the edge between needs to be similar to the feature of the edge between . Node feature similarity can also be defined in the same vain as edge feature similarity.

Given the above criteria, FINAL defines the following objective function:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | |  |
|  | |  |  | |
|  | |  |  | |
|  | |  | Eq. . | |

where is the edge attribute matrix, is an indicator function which returns 1 if the condition in the parentheses is correct and is a normalization factor. By minimizing this objective function, the alignment matrix can be found.

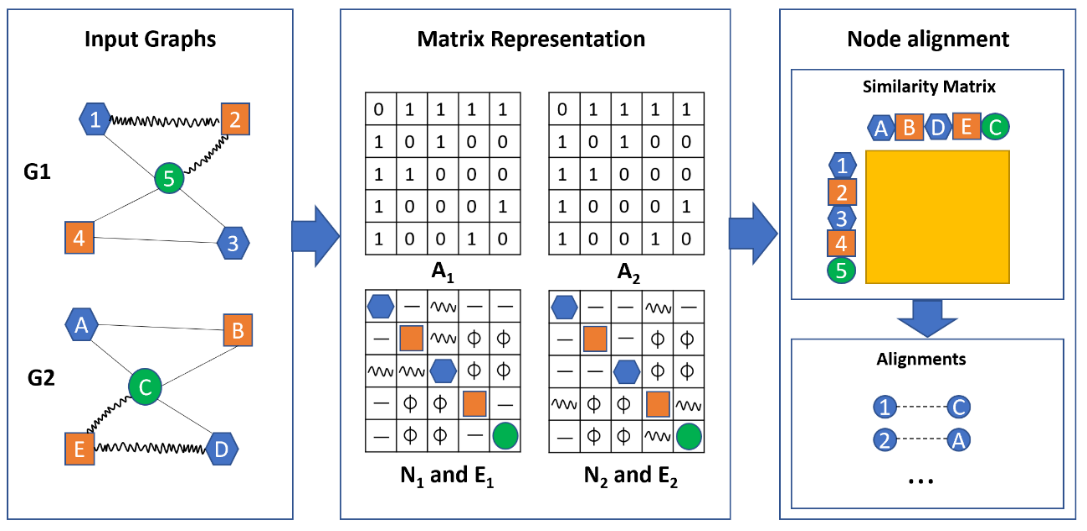


Figure 2: Overview of FINAL algorithm

In order to minimize the above objective function, FINAL rewrites it as follows:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where can be obtained by vectorizing the alignment matrix, is the symmetric normalized matrix of and .

In order to consider a prior alignment matrix , Eq. 2.12 can be rewritten as follows:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where is the vectorized version of the prior alignment matrix .

Then, the alignment matrix can be found by taking the derivative of the above objective function and setting it to 0.

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

This is equivalent to

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

From this equation, can be found using iterative procedure.

### Embedding driven techniques

A new trend of spectral method is to apply low-rank matrix factorization techniques to accelerate the obtain of embedding matrix for network nodes, then use the node embeddings to determine the alignment result. These techniques mitigate the incompatibility problem between spectral methods and large-scale networks. Here we introduce REGAL as a latest and typical example for this trending.

**REGAL.** REGAL is similar to IsoRank as it also aims to compute the alignment matrix directly. However, it takes into account both the structural similarity between the neighborhoods and the features of the nodes as well. The similarity between a node and is defined as follows:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where is the neighbor degree vector of node which captures the neighborhood of node while is the feature vector of node and is a distance function on the feature vectors. It is worth noting that REGAL computes the similarity between two nodes for both pairs of nodes that are within and across graphs. Depending the attribute type such as real-valued or categorical, the distance between two feature vectors can be their Euclidean distance or the number of different elements. On the other hand, let be the number of nodes at steps away from that have degree . Then, the -th element of is its weighted average where is a discount factor based on the number of steps from .

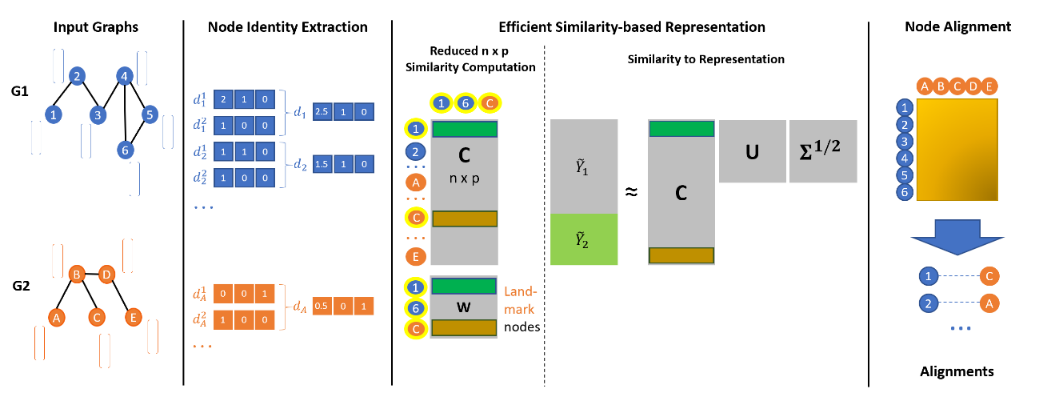


Figure 3: Overview of REGAL algorithm

Although it is possible to compute following Eq. 2.16, the number of necessary computations is which is extremely high for large graphs. To this end, REGAL proposes a method to compute efficiently using low-rank approximation. The general idea is that instead of computing the similarity between every pairs of nodes in and, REGAL aims to compute the similarity between all nodes in and and “landmark” nodes. In other words, can be approximated as follows:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where and are matrices of size . Let be the similarity matrix between and the landmark nodes while be the similarity matrix between the landmark nodes. REGAL shows that can also be computed by:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where    is the pseudo-inverse of which has the following singular value decomposition   . This means the matrix can be computed as follows:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

Then, the alignment matrix can be computed as .

## Representation learning methods

Representation learning methods approach the network alignment problem by leveraging advances in graph embedding. It involves two steps which are *embedding generation* and *alignment matrix generation*. In the first step, given a graph , a graph embedding technique is used to learn a embedding function that represents each node in by an embedding where is the embedding size (). After this step, we obtain two embedding matrices for graph respectively. Given the embeddings obtained in the first step for both graphs, the alignment matrix generation aims to learn a mapping function that maps the embeddings in to the embedding space of . Then, the alignment matrix can be learned by considering the similarity between and . More precisely, given a node and , the similarity between them is measured by where is a similarity measure. Representation-learning-based network alignment techniques differ in the embedding function and the mapping function . In the following, we discuss these methods and their function choices in detail.

### First-order proximity based techniques

The first-order proximity in a network is the local pairwise proximity information between any two nodes of the network. Most of representation learning methods consider this information to efficiently map the network nodes to the vector spaces so that the any two-node having direct connection will have closer representation. Once the topologies of the networks are well captured in the representation space, the unexpected structural noises will be reduced and thus facilitate the reconcilement step. For this kind of techniques, we present PALE as one success example.

**PALE.** PALE involves a preprocessing step in which prior mappings between two graphs are used to fill in “missing” edges that are available in one graph but not the other. The general idea is that if there is an edge between in and there are two mappings between and , there should be an edge between if this edge is not already available.

*Embedding function:* Given an “extended” graph, PALE constructs a graph embedding function as follows:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

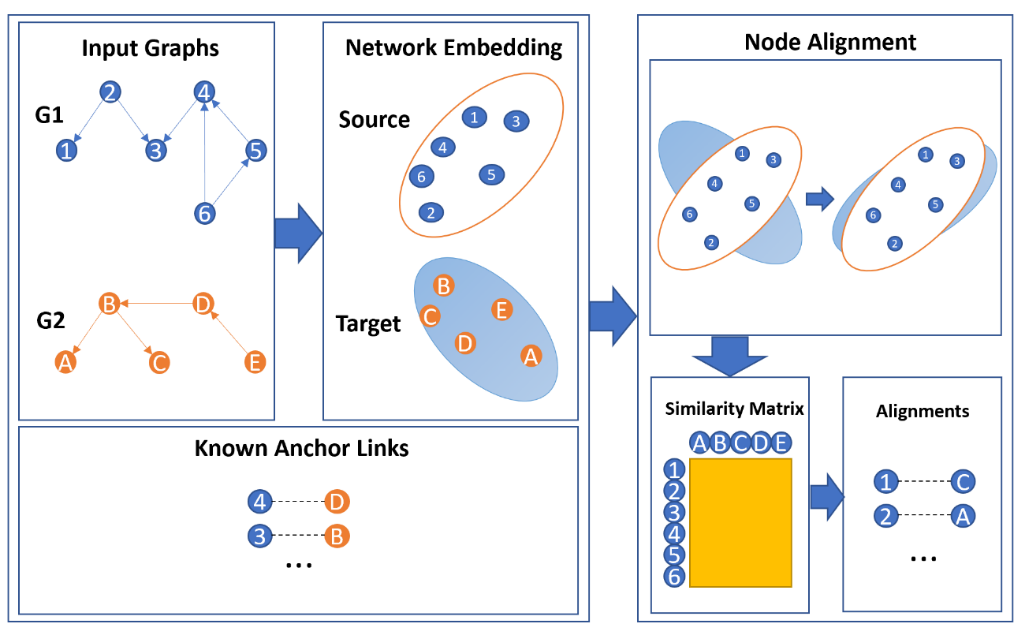


Figure 4: Overview of PALE algorithm

where is and embedding matrix of size and denotes the -th row in the matrix which corresponds to the embedding of node . The elements of the embedding matrix can be learned by minimizing the following loss function:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where is an observed edge and is a negative node that is sampled following a negative sampling strategy . The above loss function captures the idea that the distance between connected nodes (e.g. ) should be high. In the above loss function, the distance between two nodes is measured by their dot product . The negative nodes are used to prevent a trivial case in which all the embeddings are mapped to the same point. Given the loss function , the embedding matrix can be learned using stochastic gradient descent.

### Second-order proximity based techniques

The counterpart of the first-order proximity is second-order proximity, which reflects the similarity if any two nodes through the similarity between their neighborhood network structure. In other words, second-order proximity captures more global proximity information in comparison to first-order proximity. In actual practice, instead of using directly the edge-list of the graph, the second-order proximity based technique often employs random walks to construct the node contexts. Although these techniques seem to create more structural noise than their counterpart, they still achieve some remarkable result, especially when sophisticated reconcilement strategy are employed on the learned representation spaces. We here present DeepLink, a success case, to illustrate for this technique class.

**DeepLink.** While DeepLink obtains the same approach in constructing the graph embedding as PALE, its mapping function differs as it considers the mapping direction as well.

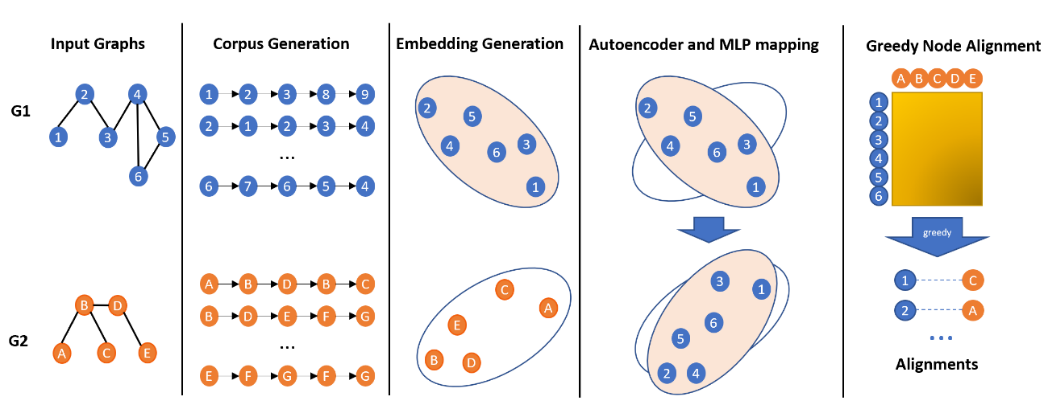


Figure 5: Overview of DeepLink algorithm

*Embedding function:* Similar to PALE, the embedding function is an embedding matrix:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

In addition, the loss function DeepLink uses to find the parameters of **E** is also similar:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

However, while PALE considers two nodes to be similar if they form an observed edge, DeepLink relaxes this constraint by considering two nodes to be similar if belongs to some random walks starting from .

*Mapping function:* DeepLink define the mapping function from to as follows:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

Similarly, the mapping function from to is: . The parameters of the mapping function can be found by optimizing the following loss function using SGD:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

The parameters of can be obtained in the similar manner.

Given two mapping functions and , DeepLink aims to improve their quality using *Linkage Dual Learning* process which involves unsupervised and supervised learning steps. The *unsupervised learning step* is similar to the auto mapping process in an autoencoder [19] in which the embedding of a node in and its embedding after applying two mapping functions and should be similar. Formally, we want to minimize the difference between and . This means the parameters of and can be improved by minimizing the following loss function:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

The *supervised learning step* aims to improve the quality of the mapping functions further by leveraging the anchor nodes in a dual learning process. Given a node and its corresponding node , the embedding of in () and the embedding of (**v**) should be the closest among all node embeddings of . However, as it is difficult for the embedding of to be the closest to , DeepLink considers the top-k closest to instead. Formally, given a node and its corresponding node , in order to learn the parameters of , DeepLink aims to maximize the following reward function:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where is the top-k nodes in that have embeddings closest to . Similarly, the reward function for is:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

DeepLink aims to maximize these reward functions in an alternative manner to obtain the parameters of and .

### Hybrid techniques

A natural but effective idea is to exploit both first-order and second-order proximity in embedding step so that the representation learning space can capture both local and global pairwise similarity information. We categorize these kind of techniques as *hybrid techniques* and introduce IONE, one of the typical work for this technique family.

**IONE.** Although IONE uses the same mapping function as PALE, its embedding function is more complicated as it takes into account the neighborhood of a node in computing its embedding.

*Embedding function:* Although the embedding function of IONE is also an embedding matrix, this matrix contains more elements as each row is a concatenation of three vectors: a node vector, an input context vector and an output context vector. Formally, the embedding function of IONE is:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where . For ease of notation, we denote the node vector, input context vector and output context vector of a node as respectively. The general idea of IONE is that the node vector of one node can affect the input context vector of its neighbors while the node vectors of its neighbors can contribute to its output context vector.

In order to learn the parameters of , IONE aims to satisfy two objectives: i) nodes that are close in each graph should have similar node embeddings and ii) nodes that have close embeddings are good candidates for alignment.

The first objective can be obtained by considering the probability that a node vector contributes to the input context vector of one of its neighbors in comparison with other nodes which is defined as follows:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

Its empirical counterpart is defined as follows:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where is the weight of the edge . Similarly, the probability that a node vector contributes to the output context vector of its neighbor and its empirical counterpart are defined as follows:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |
|  |  | Eq. . |

Intuitively, the probabilities should be similar to its empirical counterparts as they both model the probability that a node vector contributes to the context vector of its neighbor. Satisfying this intuition also allows IONE to satisfy the first objective, which can be captured by the following objective function:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

In order to capture the second objective, IONE considers the following observation: if and are two corresponding nodes then and are interchangeable in two social network. In other words, the node vector of can contribute to the context vectors of the neighbors of in and similarly, the node vector of can contribute to the context vectors of the neighbors of in . However, the contribution of the node vector of to the neighbors of is conditioned by the strength of the connection between and which is defined by the probability. Given these observations, the objective function is modified to consider the information regarding corresponding nodes as follows:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

Given two objective functions, the parameters of the embedding function can be found by minimizing the combined objective function: using SGD.

*Mapping function:* The mapping function of IONE is a special case of the mapping function of PALE in which the mapping matrix is an identity matrix.

# PROPOSED TECHNIQUE

In this chapter, based on the challenges of the problem that we have discussed so far, we will propose a fully unsupervised network alignment framework based on a multi-order embedding model. Given the state-of-the-art representation learning models on graphs, GCN, with the multi-order nature, we can easily map the nodes of graphs into low-dimensional vector space. Our model captures not only structural information but also prior node attribute. We further design a refinement mechanism which enable our model to be more robust to consistency violations and noise.

## Model and Problem Statement

### Attributed network

An attributed network (or graph) is a data structure that can be represented as , where is a set of nodes. is and adjacency matrix representing connections/edges/relationships in a graph; i.e. if there is an edge from to and vice versa, and otherwise. is a node attribute matrix, assuming the number of nodes is and each node has and -dimensional attribute. Each row in the attribute matrix F encodes the semantics of the respective node (e.g., age and marital status of users in social networks or category of protein in protein-protein interaction networks). Because node attributes originate from not synthetic features produced by embedding techniques or hand-crafted by experts but the application domain, node feature and graph structural information are independent.

Table 1: Notation Summary

|  |  |
| --- | --- |
| **Symbols** | **Definition** |
|  | An attributed network  Adjacency matrix with self-loop  Set of neighbors of node includes itself  Diagonal degree matrix with |
|  | Number of GCN layers  Embedding dimension at -th layer  Weight matrix at layer  Node embeddings at layer |
|  | Alignment matrix |

### Network alignment

Network alignment is the task of identifying corresponding nodes between two different networks. Without loss of generality, we select one network as the source network, , and the other one as the target network, . For each node in the source network, we aim to recognize, if there are any, its counterparts in the target network.

**Alignment matrix.** Network alignment techniques calculate an alignment matrix , where and are the numbers of nodes in and , respectively. represents the matching degree between and .

**Anchor Links.** Given two networks and , a node pair , with is an anchor link, if is aligned to . We refer to and as anchor nodes. The goal of network alignment is to produce all potential anchor links, given that little or no ground-truth information on anchor links is available.

**Problem Definition (Alignment matrix computation)** *Given two attributed networks and , the problem of network alignment is to calculate an alignment matrix where represents the matching degree between a node and .*

The above definition brings us several benefits. The first advantage is that it enables *scalable* calculation, since anchor links can be inferred form the alignment matrix through various ranking rules [5], [49]. The second advantage is that the alignment matrix enables us to return *flexible* alignment result because node pairings can be one-to-many, which is important for differently sized networks [50], [48], [51].

### Alignment constraints

As we have mentioned in the previous section, there are two types of constraints that state-of-the-art models often rely on to guarantee that these frameworks return good performance [5], [1].

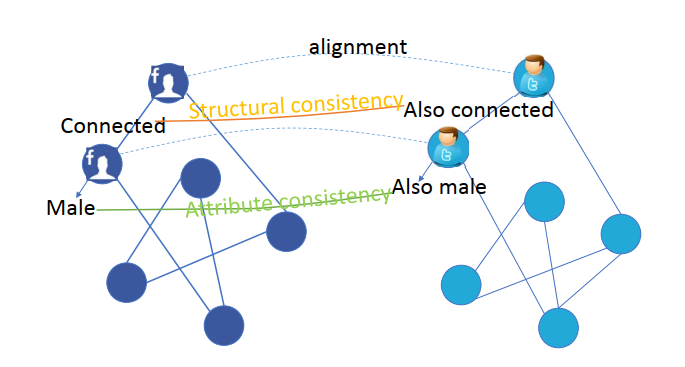


Figure 6: Structural and attribute consistency example

**Structural consistency.** This constraint is also known as the : If two nodes have close topological relation in one network, their anchor nodes in the other network shall maintain this relation [4]. For example, if two persons are friends in one social network, they are likely to connect in another social network (Figure 6). Mathematically, if are close neighbors in and and are anchor links, then shall be close in .

**Attribute consistency.** This constraint states that corresponding nodes shall share the same attribute values [5], [1]. For example, the profile information (e.g., age, email address) of a single person in different social networks shall be similar (Figure 6). Formally, if is an anchor link, then it should hold that (implying ).

Most existing alignment techniques cannot satisfy these two consistency constraints at the same time, though, due to their differences in modality [1]. While there are a few notable exceptions, they still consider both types of consistencies in separate steps [5]. In this paper, we show theoretically and empirically that the properties of GCN can be used to enforce structural consistency and attribute consistency simultaneously. Moreover, we show that, in practice, these consistency constraints may sometimes be violated, so that the model shall be made robust against minor violations.

## Approach overview

This section presents an overview of our overall approach to tackle the problem. Real-world networks often have a variety of node types, node attribute, and complex structure. Therefore, constructing the potential matching between two networks is a task of challenge. 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.

### Motivation

As the nature of real-world networks [2] and empirical results of state-of-the-art models, we highly emphasize that applicable network alignment framework should consider the followings research questions:

1. *Consistency*: As we have mentioned several times in this thesis, structural consistency and attribute consistency should be hold for network alignment since these constrains enable us to address right correspondence nodes across networks [5], [1]. Downstream applications such as personalized advertisement and friend suggestion could be badly affected by false positives matching. But with our setting defined in 3.1.2, the false negatives will be removed because the output returns an anchor for every node.
2. *Adaptivity*: In most cases, there are the violations of structural consistency and attribute consistency between source network and target network. For example, topology perturbations that violate structural consistency are common in social networks, two persons may be friends in one network but not the other. Attribute noises are common as well, e.g. social network users can register multiple accounts with different or missing attribute values due to typos, asynchronicity, laziness, privacy, or platform incompatibility.
3. *Unsupervised*: A significant number of existing alignment frameworks require large training data. However, labelled dataset with ground-truth information is rarely available because the labelling process in real-world datasets is a task of challenge due to the need of human effort [47]. For instance, creating ground-truth information for accounts across social network is a high time-consuming process because we need to check the users’ background and paring these accounts manually one by one. Therefore, unsupervised network alignment is a crucial need.

We need to overcome several challenges to fulfil these guidelines. The first thing that an aligner needs to satisfy is that it will overcome the structural as well as attribute noises. In the ideal situation, target network may be the source network with permuted node identities [53]. Second, current representation learning techniques often embed two networks independently that may introduce noises into graph embeddings, in turn makes the embeddings of two graph are in two separated vector spaces. Hence, those models need an addition reconciliation step, which is however still an approximate process [54].

### Solution Sketch

In this work, we go beyond the state-of-the-art by developing an end-to-end network alignment framework, where the source and the target networks are embedded by a GCN-based model while satisfying consistency constraints simultaneously. We provide theoretical motivations of using GCN-based model to unify the attribute and structural information of the network nodes by vector representations, where the vector-similarity of two anchor nodes depend on both the topological and semantic properties. Moreover, we design a consistency loss to guarantee that the learnt embeddings satisfying (R1).

Thanks to GCN properties, we show that the network permutation does not affect the embedding of each network node. We train the same GCN-based model for both source and target networks by a weight-sharing mechanism. This allows their embeddings to be in a common space without any labelled data, avoiding the reconciliation step and maximizing the noise tolerance of the model, satisfying (R3).

### End-to-end network alignment framework

Figure 7 presents an overview of our model. We first forward the source and target network through a GCN-based model to embed network nodes to feature spaces, in which embeddings are hidden features capturing all structural and attribute information. With this structure, our model requires the realization of the following functionalities:

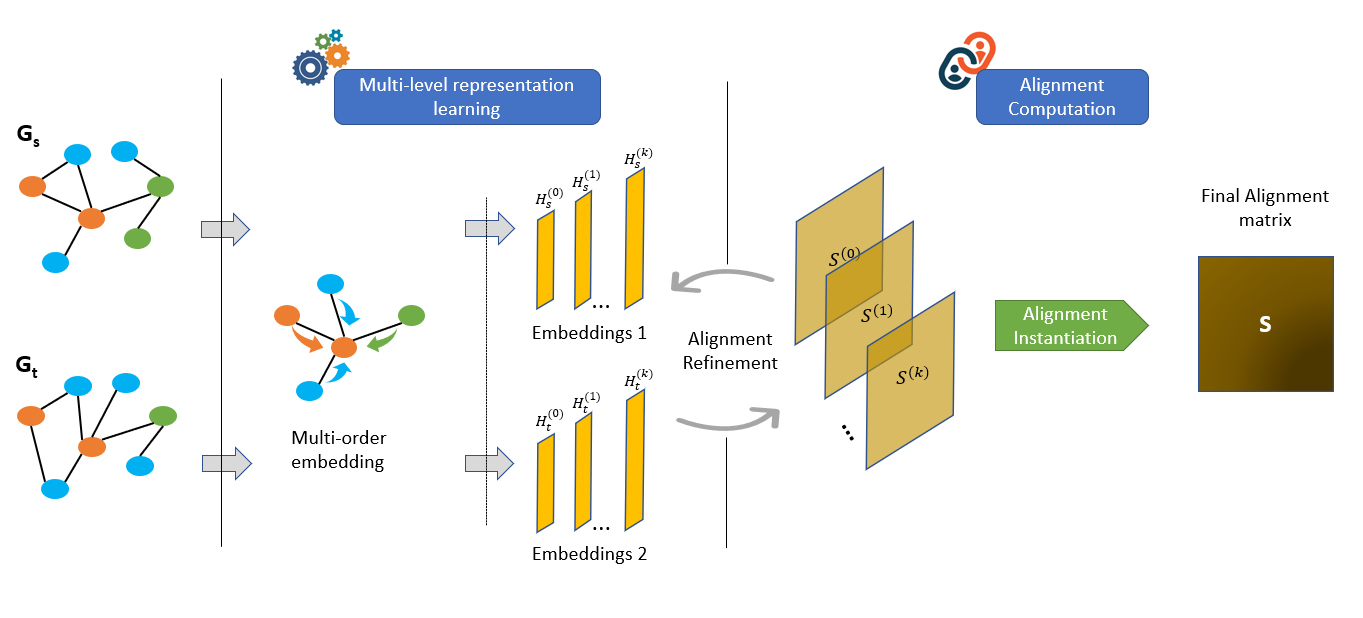


Figure 7: Overview of GAlign framework

**Multi-order embedding.** In this step, we design a specific GCN-based model for learning the representations of network nodes. The model contains several layers; each layer learns hidden features at a different order of neighborhood structure for network nodes (deeper layer, larger neighborhood). The model is guaranteed to satisfy consistency constraints via a *consistency* loss function. The detailed process of constructing these multi-order representations is described latter.

**Alignment Refinement**. Before producing the final alignment matrix, we develop an iterative refinement process to avoid the effects of real-world noises. First, we determine the stable nodes (nodes with no noises) by comparing the current embeddings across GCN layers. Then, using the current alignment output as a reference, we adjust the embeddings by amplifying the influence of stable nodes over time.

## GCN for consistent network alignment

In this section, we provide theoretical motivations to design a specific GCN model for network alignment, in which GCN properties guarantee alignment consistency constraints.

### GCN model

GCN is a deep neural network model of layers, denoted as where , that allows end-to-end representation learning of graph nodes [5]. Given node attributes and the structure of a network , GCN encodes simultaneously these two types of information into the hidden features at each layer, which can be used as individual information of the network nodes. Formally, the feed-forward pass is: , where the layer produces the embedding GCN employs a neighborhood aggregation scheme across all layers: the hidden features of the previous layer construct the features of the next layer:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where is the normalized Laplacian matrix, is the embedding matrix is the embedding dimension a t layer is the adjacency matrix with self-connections; is the identity matrix, is a diagonal matrix that  , is a trainable weight matrix at layer , and is an activation function. is not suitable for alignment task since it does not distinguish negative values because it returns zero values for all negative entrances. Since activate function has to be bijective, we use *tanh* to avoid the loss of information.

### Permutation Immunity

The target network can be modeled as a permuted version of the source network [21]. Given the adjacency matrices of source network and the target network , these works aim to find the permutation matrix that minimize the cost function which captures the consistency constraints:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

However, this approach is prone to consistency noise and restricted to 1-1 alignment. We show that using GCN can achieve similar or better with more flexibility.

**Proposition 1** *Embedding generated by GCN is immune to the permutation factor. Given for a particular permutation matrix P, we prove that and are the embedding matrices at the layer l of the model of source and target network, respectively.*

*Proof:* First, we already have , as the embeddings at the first layer of network nodes are their original attribute. Suppose that for layer , is true, we will prove that . We have:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

Because of permutation assumption, we have and . The Eq. 3.3 is then equivalent to:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

As is an orthogonal matrix, . We can rewrite Eq. 3.3 as: .

### From consistency to embedding

We prove that GCN is not only immune to permutation but also naturally supports the structural and attribute consistency constraints. In other words, two anchor nodes which share the same attribute and topology information (i.e. node degree, neighbors) will have same embeddings at every layer.

**Proposition 2** *If two nodes , have save degree and there exists a matching between the neighbor sets of two nodes such that for every matched pairs :*

* *Degree is equal: deg(t) = deg(t’)*
* *Embedding at the layer l is equal: . Them embeddings of and at layer are the same, which means:*

*Proof:* At layer the embedding of and is:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |
|  |  | Eq. . |

where . Since and , we have .

## Multi-order embedding

In this section, we design a specific GCN model to embed the nodes of the source and target networks into a unified vector space, in which the embeddings of true anchor nodes are similar, allowing us to retrieve the alignment result efficiently.

### GCN-based embedding model

The model contains layers; each layer learns hidden features at a different order of neighborhood structure for each network node. Starting from the lowest embedding that presents the node attributes, the model iteratively aggregates the hidden feature of every node in one layer with features of its adjacent nodes to produce the hidden feature for this node in the next layer and so on (see Eq. 3.1). This message-passing scheme allows the model to unify attribute information with topology information at different orders [56].

Existing works use only the embeddings in the final layer of GCN for representing the network nodes. However, there is a trade-off: deeper layers contain more structural information but prone to structural noises and vice-versa (shallower layers focus on more attribute consistency but prone to attribute noises). Indeed, the hidden features can be considered as the collective information of -hop neighborhood of nodes and thus, a larger neighborhood would hinder individual information of each node.

For this reason, we leverage embeddings at all layers. Formally, each node is associated with a set of embeddings , or so-called multi-order features.

Using all layers leverage the structural information locally and globally (from short-range to long-range neighborhood).

However, the number of GCN layers is still a hyperparameter and we show how to choose a suitable in the experiments.

### Consistency loss

To make the embedding satisfy the consistency constraints (see (R1)), we design a *consistency* loss function that encourages the nodes that have similar neighborhood structure to have similar embeddings while making those of unrelated nodes highly distinctive from each other. On the one hand, lower-order embeddings (shallower layers) capture local information of a node such as neighborhood structure. However, since there could be many nodes with the same local information in a network, the lower-order embeddings of different nodes are often similar even if they are far away from each other or, making the alignment difficult. On the other hand, high-order embeddings might collapse the embedding distribution.

Embedding at deeper layers can be seen as the result of averaging large number of nodes into one vector, leading to similar embeddings due to the law of large number. Nevertheless, taking neighbors within a large number of hops, means the information of many nodes can appear in each other’s context, pulling their feature vectors closer in the embedding space. Balancing this trade-off, we compute the loss function from the embeddings at all layers to complement each other:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where denotes the Frobenius norm. We use normalized Laplacian matrix instead of adjacency matrix with the aim to enrich the embeddings with more topology information, which avoids collapsing the embedding space.

### Put It Altogether

|  |
| --- |
| **Algorithm 1** Learning for multi-order embedding |
| 1: **Input**: Source network , target network  2: **Output**: Multi-order node embeddings of and  3: Construct GCN model  4: **for**  in **do**  5: Initialize layer-0 embeddings  6: **for** some epochs **do**  7: **for**  in **do**  8: Compute multi-order embeddings for by Eq. 3.3  9: **for**  in **do**  10: Compute the loss function by Eq. 3.7  11: Update parameters by back propagation  12: **return** and |

Alg. 1: Learning for multi-order embedding

Alg. 1 illustrates the whole multi-order feature construction process. The core idea of our training algorithm is the weight-sharing mechanism, in which the GCNs of the source network and the target network use the same weight matrix at every layer . Otherwise, the learnt embeddings of different networks will end-up in different embedding spaces, breaking the alignment purposes in the first place. Moreover, the weight-sharing mechanism guarantees that the consistency constraints are satisfied, as proven in 3.3.

## Alignment Computation

In this section, we show how to exploit the learnt multi-order embeddings to compute the alignment matrix effectively.

### Alignment Instantiation

As aforementioned, the embeddings at different GCN layers represent featural information of each node at different topological order. Shallow layers tend to handle local neighborhood information, while deeper layers tend to capture global network topology. Combining all layers simultaneously is beneficial as true anchor nodes should have not only similar local features but also similar global features.

However, there are issues to implement this idea. First, the layers could be not equally important. For example, the first layer carries neighborhood structure of each node, but easily suffer from structural noises. Deeper layers are more robust to structural noises since they consider a larger range of topology information, but might be less useful for aligning individual nodes. Second, the node features of the source network and target network might not have the same embedding space. Existing alignment frameworks often perform an additional reconciliation step, which however still have discrepancy and cause instability of alignment [57], [4], [54], [58].

**Layer-wise alignment matrix.** To mitigate these issues, we first define the layer-wise alignment matrix aggregated form the embeddings at each layer :

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

As the weights of the GCN models are shared between the two networks, their embedding spaces are the same and thus their alignment can be measured directly by the analogy of their embeddings. Each layer-wise alignment matrix represents the alignment scores for all pairs of nodes between the source and target networks. In other words, it captures, for each node, the alignment candidates according to all embedded structural and attribute information at the current layer.

**Aggregated alignment matrix.** We aggregate all layer-wise alignment matrices into a single alignment matrix:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where is the importance of layer and we treat them as hyper parameters. For one-to-one network alignment setting, the anchor links can be instantiated straightforwardly by taking the top-1 target node (highest alignment score) for each source node. Other alignment settings such as one-to-many can be instantiated as well, but out of the scope of our paper.

### Alignment refinement

Another issue when computing alignment from embeddings is the effect of adversarial conditions (or noises, for short), which refers to structural and attribute noises as well as constraint violations and graph size imbalance. Intuitively, nodes with noises will bring instability in their embeddings over neighboring nodes. As a result, noises will make the embeddings of true anchor nodes different, leading to incorrect alignment scores. However, the vice-versa is not true since the embedding difference is only caused by noises and we do not know the true alignment before-hand. We design an iterative refinement process to detect and mitigate the effects of noises on the aggregation of embeddings into alignment matrix.

**Stability of embeddings.** The first step is to evaluate the stability of node embeddings. As aforementioned in $IV-B, the target network can be considered as a permuted version of the source network with the noises added. Since GCN embedding is immune to permutation (Prop. 1), the embedding difference between two true anchor nodes is only caused by consistency noises. Thanks to this property, we define stability as follows. Given a node and their corresponding embeddings at all layers , we want to determine if node is a stable node or not. A node is defined as stable if its anchor nodes (highest alignment scores of ) in all layer-wise alignment matrices are the same and the corresponding alignment scores must be higher than a confident factor :

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

The motivation behind this formulation is that if a node is stable, its anchor node should be stable as well. In other words, a pair of anchor nodes is stable if they have similar embeddings at any layer of the model. On the other hand, nodes that do not satisfy Eq. 3.10 are called unstable nodes.

**Noise-aware propagation of embeddings**. In terms of GCN aggregation, we do not want unstable nodes to aggregate their information into neighboring nodes. We propose a weighted propagation mechanism for GCN embeddings such that more stable nodes have greater influence and vice-versa:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where and . is the influence factor of node . Intuitively, if is a stable node, its influence should be intensified:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where is an accumulation constant. Put it altogether, the new aggregation rule of GCN is:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where and is a diagonal matrix ().

|  |
| --- |
| **Algorithm 2** Alignment computation with stability refinement |
| 1: **Input: and : source and target graphs**  Source embeddings  Target embeddings  2: **Output**: Fine-tuned alignment matrix  3: Compute layer-wise alignment matrices from embeddings by Eq. 3.8  4: 🡪 Initialize all influence factors to 1  5: Initialize  6: **for** some iterations **do**  7: Find stable nodes and using Eq. 3.10  8: f**or** , **do**  9: Update and using Eq. 3.10  10: Update embeddings  11: Update from  12: Compute aggregated alignment matrix  13: **if** **then**  14:  15: **return** |

Alg. 2: Alignment computation with stability refinement

**Alignment Algorithm.** The whole process of alignment algorithm is put altogether as Alg. 2. At the beginning, we aggregate the embeddings into layer-wise alignment matrices as in 3.5.1. Then, the alignment is refined by a searching strategy. First, we initialize the influence factor of each node to 1. Next, an iterative process is performed. On the one hand, we use the current alignment matrices as reference anchor links (highest alignment scores) to detect stable nodes. The influence factors of those nodes in turn are increased. On the other hand, those factors are used to refine the current embeddings, which is then aggregated into new alignment matrices. However, the search space of this iterative process is not uniform since there is no supervision information. We employ a greedy selection based on the criterion that true anchor nodes should have embeddings close to each other as much as possible. More precisely, we keep track the sum of top-1 alignment scores for each refined alignment matrix, i.e. , and return the one with best score in the end.

### Complexity Analysis

Without loss of generally, we rely on: as the number of nodes, as the number of feature dimensions, and as the number of edges (non-zero entries of adjacency matrix).

**Time Complexity.** There are two steps to analyze:

* *Multi-order embedding:* The normalized Laplacian matrix is computed once. Since the adjacency matrix is sparse and is diagonal matrix, the time complexity for calculating C is . Therefore, the propagation of GCN layers takes .
* *Alignment computation:* First the refinement process relies on the forward pass, which takes as above. Second, finding stable nodes take . Updating the influence factors of stable nodes in worst case is (all nodes are stable).

Therefore, the total time complexity is .

**Space Complexity.** There are also two steps to analyze:

* *Multi-order embedding:* First, we need to store embeddings at every layer, which takes . Second, we need to store trainable parameters at every layer, which takes . Third, the normalized sparse Laplacian matrix takes space complexity. Put it all together, the space complexity of embedding step is .
* *Alignment computation:* We do not need to store the whole alignment matrix in memory. All operations we use on is finding stable nodes, which can be done by separately iterating the rows of . In other words, we only need to compute and store one-row vector of from the embeddings at each iteration, which takes space.

In sum, the total space complexity is .

# EXPERIMENT EVALUATION

Our experiments answer the following research questions:

1. Does our model outperform the baseline methods?
2. How does each component of our model matter?
3. Is our model adaptive to adversarial conditions?
4. Is our model sensitive to hyper-parameters?
5. Can our technique be interpreted qualitatively?

In the remainder, we first describe our experimental setting (4.1). Then we present our empirical evaluations, including end-to-end comparison (4.2), several ablation tests (4.3) adaptivity to adversarial conditions (4.4), and hyperparameter sensitivity (4.5).

## Experiment setup

**Real datasets.** We use three state-of-the-art alignment datasets of 6 real-world networks with different domains.

* *Douban Online vs Douban offline:* Douban network is a Chinese social network with nodes as users and edges as friendships [59] containing 1118 anchor links [1].
* *Flickr vs Myspace:* The alignment ground-truth between subnetworks of Flickr and Myspace is extracted and validated by [1], containing 323 anchor links.
* *Allmovie vs Imdb:* Allmovie network is constructed from Rotten Tomatoes[[1]](#footnote-1). Two films have an edge connecting them if they have at least one common actors. **Imdb** network is constructed in a similar way from Imdb[[2]](#footnote-2). The alignment output is constructed by the identity of the film, containing 5176 anchor links.

Table 2: Statistics of real-world networks

|  |  |  |  |
| --- | --- | --- | --- |
| **Networks** | **#Nodes** | **#Edges** | **#Attributes** |
| Douban Online  Douban Offline  Flickr  Myspace  Allmovie  Tmdb  Bn  Econ  Email | 3906  1118  5740  4504  6011  5713  1781  1258  1133 | 8164  1511  8977  5507  124709  119073  9016  7619  5451 | 538  538  3  3  14  14  20  20  20 |

**Synthetic data.** We further synthesize alignment data to comprehensively evaluate adversarial conditions such as noises or graph size imbalance. The adversary level is controlled by a synthesis procedure similar to $V-C. For an original network, we generate a noisy version and perform network alignment between the two versions (node identity is preserved, indicating the alignment ground-truth). We apply the following real-world networks as they do not have alignment data yet.

* *bn*: represents a part of brain [59]. Each node of the network depicts a brain voxel, and the edge represent one fiber tract that connects two voxels.
* *econ*: represents an economic model of Victoria state, Australia during the banking crisis in [60]. The nodes represent the organizations located in the state (e.g. firms, banks), and the edge represent the contractual relationships between them.
* *email*: is generated from email data of European universities [60]. The nodes represent the email addresses, and the edges between the any two addresses are formed if they both sent and received emails from each other.

Table II summarizes the networks used for alignment.

**Baseline methods.** We study five representative methods:

1. *REGAL*: is a spectral method which models alignment matrix by topology and nodes' feature similarity then employs low-rank matrix approximation speed-up.
2. *IsoRank*: is a spectral approach which propagates the pairwise node similarity over the network with the homophily principle assumption, which states that two corresponding nodes in two networks connect to similar characteristic neighbors.
3. *FINAL*: is a spectral technique which defines a model with three criteria, namely structure consistency, node feature consistency and edge feature consistency to tackle alignment problem on attributed networks.
4. *PALE*: is an embedding-based technique which learns nodes embedding by maximizing the co-occurrence likelihood of edge's vertices then applies linear or multi-layer perceptron (MLP) as mapping function.
5. *CENALP*: is an embedding-based model that unifies network alignment and link prediction tasks in a unified model, which first leverages a tailored biased random-walk strategy across the networks, then learns nodes embedding by maximizing the co-occurrence likelihood of nodes within the walks.

It is worth noting some baselines require supervision data in the form of *prior alignment matrix* (*FINAL* and *IsoRank*) and partial ground truth to reconcile embedding spaces (*PALE* and *CENALP*). To respect their original settings (even though our model is handicapping), we use 10% of ground truth for training (*PALE* and *CENALP*) and generating the *prior alignment matrix* if it is not available (*FINAL* and *IsoRank*).

**Metric.** We evaluate network alignment with state-of-the-art metrics in both *prediction* perspective and *ranking* perspective [62]. For *prediction* perspective, we employ *Success@q* (aka *Accuracy@q* [1]), which indicates whether the true positive match occurs in top- candidates. More precisely, for each anchor pair in the ground-truth, if the alignment score is within the -highest values in the row of the alignment matrix , the alignment output for node is recorded as a successful case:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

For *ranking* perspective, we use Mean Average Precision [4] (aka Mean Reciprocal Rank under pair-wise setting):

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

where is the rank of true anchor target in the sorted list of anchor candidates. Another popular ranking metric is AUC, which reflects the trade-off between precision and recall. In network alignment setting where the output must yield an anchor link for every node, AUC can be simplified to [45]:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

**Hyperparameter tuning.** If not stated otherwise, the hyper-parameters are set as follows: (accumulation factor of stable nodes), (embedding stability threshold), (number of GCN layers), the importance weights of all GCN layers are equal . Embedding size of all GCN layers . On the other hand, we also study hyper-parameter sensitivity in (4.5). For baseline methods, we report the best performance following the parameter tuning of their original papers.

**Reproducibility environment** The results are averaged over 50 runs to mitigate randomness. All 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. We use Pytorch for implementation and Adam as gradient optimizer.

## End-to-end comparison

**Effectiveness.** Table 3 answers (RQ1) by showing the end-to-end comparison of our alignment model against baseline methods. In general, our model outperforms all the baselines across all datasets in terms of *MAP*, *AUC*, and *Success@1* metrics. *FINAL* is the best method among the baselines and emerges to compete our model in Allmovie-IMDB dataset in terms of *Success@10*. This is because *FINAL* is a state-of-the-art method that is similar to us, takes into account structural information and node attribute information. Moreover, *FINAL* also considers the supervision data in the form of prior alignment matrix, while ours do not examine any.

Table 3: Network alignment comparison on real-world datasets

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Dataset** | **Metric** | **GAlign** | **CENALP** | **PALE** | **REGAL** | **IsoRank** | **FINAL** |
| **Douban Online -Offline** | MAP  AUC  success@1  success@10  Time(s) | **0.5632**  **0.9917**  **0.4526**  **0.7800**  10157.2 | 0.3537  0.7429  0.2572  0.4618  10157.2 | 0.1901  0.8899  0.0775  0.4479  68.1 | 0.1005  0.9107  0.0456  0.2030  **14.3** | 0.1299  0.9005  0.0903  0.2048  25.5 | 0.5539  0.9872  0.4383  0.7710  198.7 |
| **Flickr-Myspace** | MAP  AUC  success@1  success@10  Time(s) | **0.1608**  **0.9738**  **0.0774**  **0.3127**  93.1 | 0.132  0.8670  0.0687  0.2302  25520.0 | 0.0059  0.5444  0.0000  0.0206  123.0 | 0.0990  0.9692  0.0464  0.1950  **33.9** | 0.0085  0.6470  0.0000  0.0275  222.6 | 0.0429  0.6130  0.0206  0.0722  249.1 |
| **Allmovie-Imdb** | MAP  AUC  success@1  success@10  Time(s) | **0.8496**  **0.9971**  **0.8214**  0.9003  336.5 | 0.5693  0.9581  0.4866  0.8327  57401.0 | 0.7601  0.9868  0.6947  0.7159  1679.7 | 0.1888  0.9862  0.0953  0.6427  **76.1** | 0.5271  0.9596  0.4653  0.6427  323.7 | 0.8459  0.9885  0.7647  **0.9609**  353.3 |

A key finding is that alignment methods ill-perform significantly in Flickr - Myspace dataset. This could be explained by the fact that the networks are sparse with average degree less than five, making the structural consistency often being violated. Methods that strictly follow this consistency constraint would fail in such cases. For example, it is normal for two peoples to be friends on Facebook but not connected to each other on Twitter; or one person could use different profile attributes for different social network accounts.

**Efficiency.** The running time of alignment techniques is shown in Table 3. *REGAL* is the fastest because of low-rank matrix approximation. Our model stays in top-3 with less than 6 min.

## Ablation test

We answer (RQ2) by comparing our model and its variants:

* *GAlign-*1: The refinement step ($VI-B) is removed. The learned multi-order embeddings are used directly to compute the alignment matrix.
* *GAlign-*2: Only the embeddings at the final layer of GCN is used (as in traditional works) instead of using the multi-order embeddings.

Table 4: Ablation test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset** | **Metric** | **GAlign** | **GAlign-1** | **GAlign-2** |
| **Douban** | MAP  *Success@1* | **0.5632**  **0.4526** | 0.5622  0.4453 | 0.3467  0.2290 |
| **Imdb** | MAP  *Sucess@1* | **0.8496**  **0.8214** | 0.6894  0.6100 | 0.6934  0.6376 |

Table 5 presents the result with only important metrics and datasets due to space limitation. It can be seen that our original model *GAlign* outperforms other variants, which proves the importance of our refinement mechanisms. In particular, *GAlign* has *Success@1* is around 20% better than *GAlign-2*, which confirms the correctness of using multi-order embeddings over traditional single-order embeddings.

## Adaptivity to adversarial conditions

We answer (RQ3) by evaluating several adversarial factors.

**Structural noises.** In this experiment, we study the effect of structural noises. The noises are added by removing edges randomly. Figure 8 illustrates the *Success@1* results, where the ratio of edge removal is varied from 10% to 50%. In general, all methods suffer performance drop when the noise level increases. Our model outperforms the baseline methods, with the *success@1* goes from nearly 100% to around 80% when the edges removal ratio goes from 10% to 50%. Our model keeps the margin of 20% with the runner-up (*FINAL*). The performance of the two models *PALE* and *REGAL* drop more dramatically than the others. IsoRank does not perform well even when the noise level is low, as IsoRank may need more prior supervised information.

|  |  |
| --- | --- |
| (a) BN | (b) Econ |
| (c) Email | |

Figure 8: Robustness against structural noises

**Attribute noises.** This experiment studies the effect of attribute noises. The noises are added by changing the nodes' attribute randomly. Figure 9 depicts the results with the attribute noise level varies from 10% to 50%. We only consider *REGAL*, *FINAL* and *CENALP* as baselines because other methods do not utilize attribute information. In general, the alignment output is worse when the noise ratio increases. Interestingly, our model maintains superior performance at all levels of noise and across datasets, with the *Success@1* drops from nearly 100% to 60% when the noises go up to 50%. It is worth noting that the attribute noise has more effect on our model than the structural noise. For the baselines, *REGAL* is more robust to attribute noise than *FINAL* and *CENALP*.

|  |  |
| --- | --- |
| (a) BN | (b) Econ |
| (c) Email | |

Figure 9: Robustness against attribute noises

**Isomorphic level.** This experiment investigates whether alignment methods can be used for an orthogonal problem of reconciling different types of networks (e.g. social network vs citation network). Figure 10 depicts the result, where we vary the isomorphic level by synthesizing the source and target networks from an original network such that two of them share a ratio of original nodes as overlap. In general, the alignment performance drops when the isomorphic level is small. However, our model outperforms in all cases, with a *Success@1* margin of 30% better than the runner-up (*REGAL*).

|  |  |
| --- | --- |
| (a) BN | (b) Econ |
| (c) Email | |

Figure 10: Robustness against isomorphic level

## Hyperparameter sensitivity

This experiment answers (RQ4). Only important hyperparameters are shown due to space limitation.

**Number of GCN layers.**  Figure 11 shows the effect of the number of GCN layers , on our model performance, where column represents our multi-layer approach and other columns represent the cases of using the embeddings at that layer only. While produces the best result, it is interesting to see that using more number of layers does not increase alignment performance. This result confirms existing empirical evaluations [62], which point out the paradox that too deep GCNs are often worse than 2-layer models.

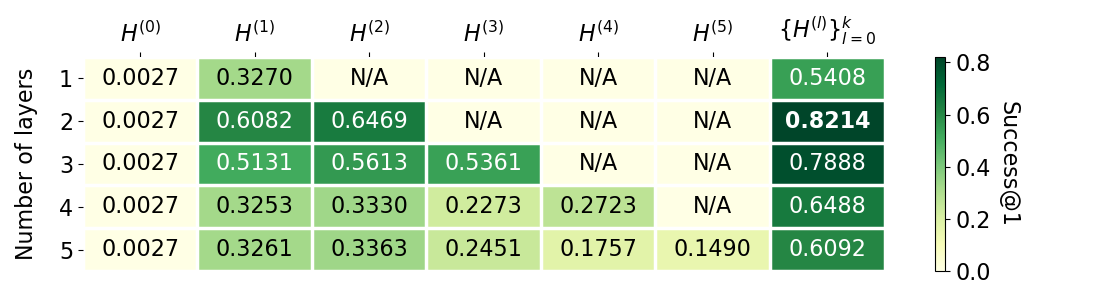


Figure 11: Effects of #GCN-layers against Success@1 (w.r.t embeddings used to compute alignment matrix)

**Important of GCN layers.** Table V shows the alignment performance with different setup of importance weights of GCN layers. Here, the weights represent the relative importance of one layer over another (their sum is 1). It can be seen that using only one layer (corresponding weight = 1) does not lead to the best performance and sometimes degrades the alignment output significantly (especially when using only node attributes). Interestingly, the best case happens when we put most of importance in the middle layer (, less importance in the deeper layer (, and more importance in the shallower layer (. This is because each network has a particular structural degree (e.g. diameter, communities, node degrees); and thus, finding the right amount of neighboring information is important.

Table 5: Layer Weights

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 0.33 0.33  0.33 | 0.33  0.50  0.17 | 0.33  0.17  0.50 | **0.8214**  0.8002  0.8179 |
| 0.00  0.67  0.33 | 0.67  0.00  0.67 | 0.33  0.33  0.00 | 0.7120  0.7820  0.7298 |
| 0.00  0.00  1.00 | 1.00  0.00  0.00 | 0.00  1.00  0.00 | 0.6082  0.6469  0.0027 |

**Embedding dimension.** Figure 12 studies the sensitivity of the embedding dimension of GCN layers. In general, users should not choose a high number of dimensions as it does not increase the performance (*Success@1*) significantly while the time and space complexity definitely become larger.

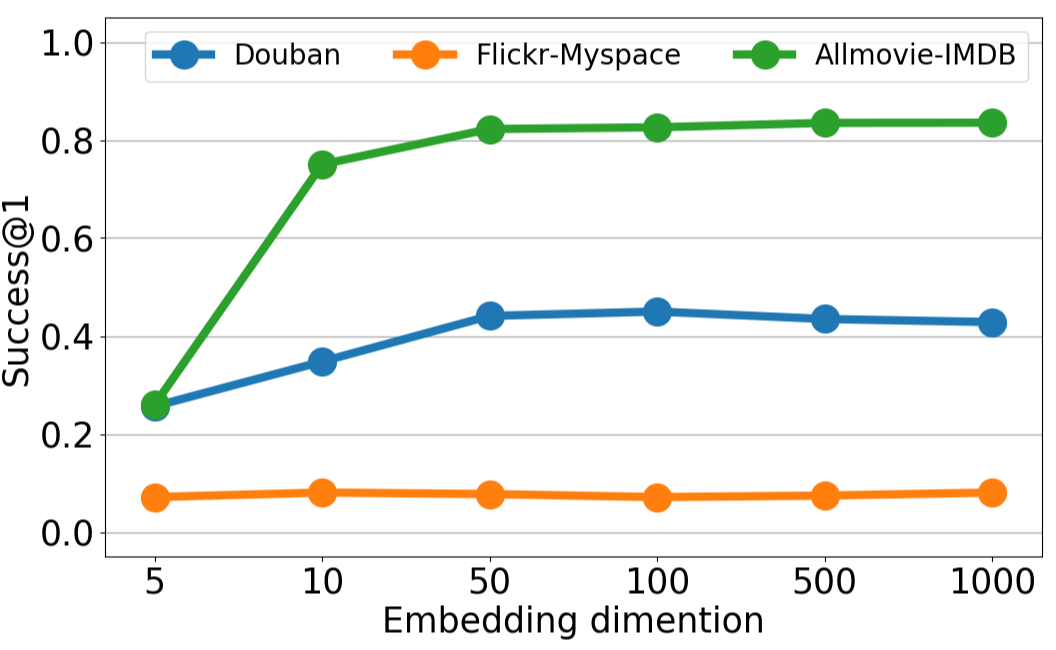


Figure 12: Embedding dimension

## Qualitative study

We answer (RQ5) by demonstrating our whole pipeline on a toy dataset, which is extracted from 10 movie pairs of the AllMovie - IMDB dataset. The node attribute is the movie categories. The multi-order embedding step is visualised by t-SNE in: (i) Figure 13a for embeddings at final layer (traditional), (ii) Figure 13b for the multi-order embeddings (the embeddings at all layers are concatenated). It can be seen that the multi-order approach produces closer embeddings for anchor nodes. The refinement step is visualised in (iii) Figure 13c, which improves the result by making the embeddings of anchor nodes more distinctive to others (e.g. embeddings of "School Ties" and "Duets" are separated after refinement).

|  |  |
| --- | --- |
| (a) Traditional embeddings | (b) Multi-order embeddings |
| (c) Multi-order embeddings after refinement | |

Figure 13: Qualitative study

# KNOWLEDGE GRAPH ALIGNMENT

To demonstrate that our model has a strong applicability in solving real-world problems, in the last subsections of this chapter, we will show our model’s performance in a real-world problem that have been attracted a large number of researches, which is *Knowledge graph alignment*.

Knowledge graph has become a popular structure to unify real-world entities by modeling the relationships between them and their attributes. To support multilingual applications, a significant number of language-specific knowledge graphs have been built by different parties using different data sources. As a result, these monolingual knowledge graphs are often disconnected, causing semantic heterogeneity and diminishing the purpose of knowledge graph in the first place. Entity alignment - the task of identifying corresponding entities across different knowledge graphs - has attracted lots of attentions in both academia and industry. However, existing alignment techniques often require many labeled data, unable to encode multi-modal data simultaneously, and enforce only few consistency constraints. In this subsection, we will apply *GAlign* with several changes so that the model can fuses different type of information to exhaustively exploit the richness of KG data. The model determines the attribute-based correlation by leveraging the advances of translation machine. Then, we adopt a late-fusion mechanism to put all the information together, which allows this information complete each other and thus enhance the final alignment result. Empirical results on real-world KGs show that our model is up to 22.71% more accurate and orders of magnitude faster than the best existing baselines.

## Knowledge graph representations

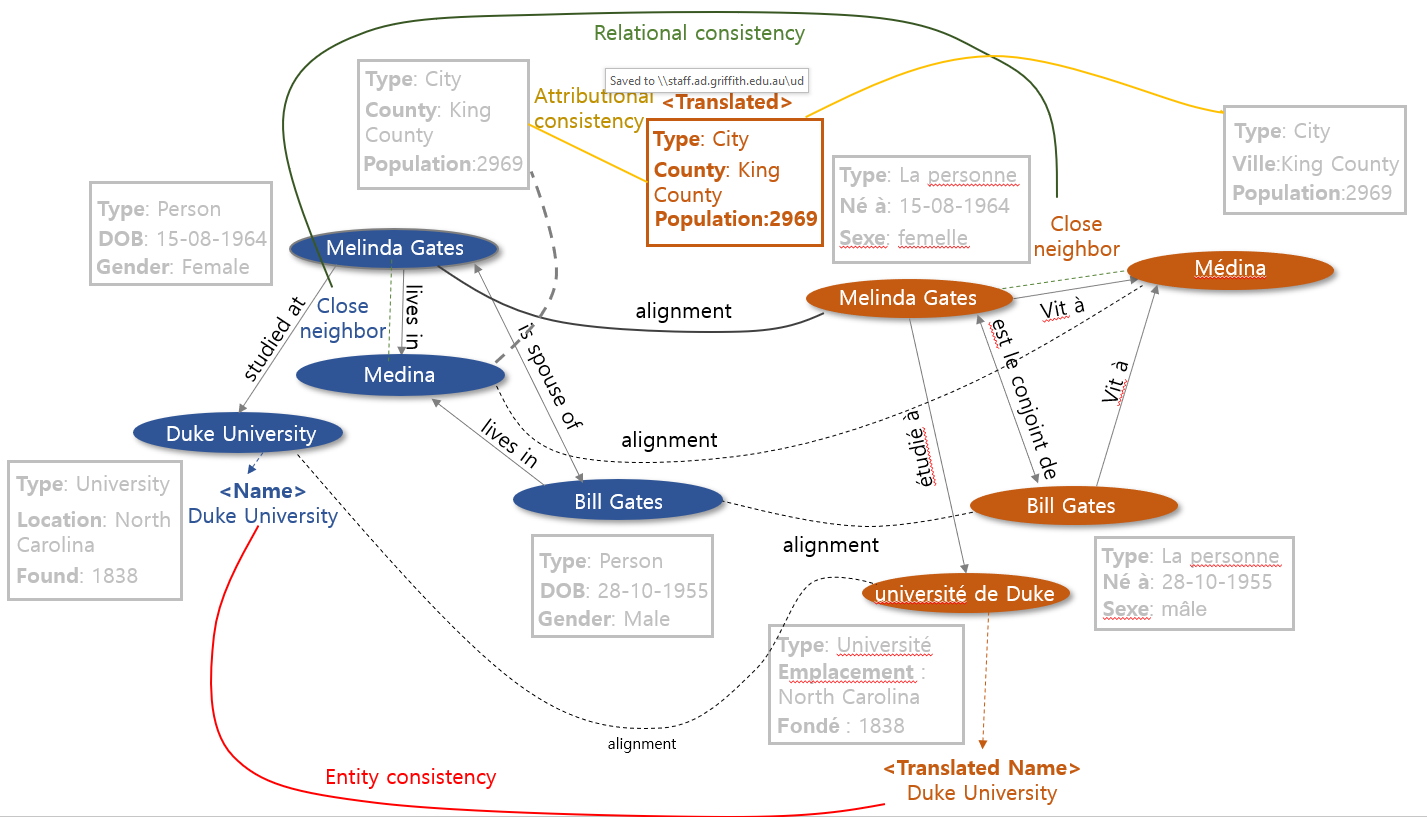


Figure 14: Example of knowledge graph entity alignment

**Triples.** The atomic unit in a knowledge graph is a triple , often abbreviated as , which depicts a relation between a head (an entity) and a tail (an attribute or another entity). There are two types of triples: (i) *relation triple* - represents the relationship between two entities, thus the structure ; and *attribute triple* - describes an attribute of an entity, hence the form . For example, in YAGO, is a *relation triple* and is an *attribute triple*.

**Knowledge graph.** On top of the above triple form, a knowledge graph can be represented as , where E is the set of entities, R is the set of relations, A is the set of attributes, V is the set of values, is the set of relation triples, and is the set of attribute triples. The size of a KG is the number if its entities.

## Knowledge Graph Alignment with GAlign

Knowledge graphs is complicate and hence to solve the alignment problem on this kind of data is not just simple as what we have done on regular datasets. Thus, instead of directly applying GAlign’s architecture to align two graphs, we first separate the alignment process into two steps:

* **Relational-aware alignment:** In this step, we first look at the relation triples of two KGs. By treating those triples as the representation of two regular graphs, we can apply GAlign to algin two sets of relation triples, in which, the node attributes are entity name. Recall that each KG are represented in different language, so we apply a preprocessing step to our data. We translate all text into English, and then using pre-trained word embedding model (Glove) to initialize prior entity attribute. Then we apply our model on the attribute graphs to return the relational alignment matrix denoted as
* **Attributional-aware alignment:** It is the fact that using just relational information does not exploit all the rich information that a KG contains. Therefore, we propose another step which makes sure the attribute information can be used to improve the alignment result. In this step, we translate both attributes as well as values of all attribute triples. Now, for each entity, we have a translated set of attributes and a translated set of value. We use simple string comparison and Jaccard similarity metric to calculate two alignment matrices that are and respectively.

After constructing and , we just combine them by using weighted sum of those three matrices to produce the final alignment matrix:

|  |  |  |
| --- | --- | --- |
|  |  | Eq. . |

## Experiment Evaluation

**Datasets.** We use three state-of-the-art datasets from DBP15K [30]. The datasets were generated from DBpedia, a large-scale multilingual knowledge base containing rich inter-language links between different language versions.

Table 6: Statistics of real-world datasets

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Dataset** | **Lang.** | **#Entity** | **#Rel.** | **#Attr.** | **#Rel.triples** | **#Att.triples** |
| **ZH-EN** | Chinese  English | 66,469  98,125 | 2,830  2,317 | 8,113  7,173 | 153,929  237,674 | 379,684  567,755 |
| **JP-EN** | Japanese  English | 65,744  95,680 | 2,043  2,096 | 5,882  6,066 | 164,373  233,319 | 354,619  497,230 |
| **FR-EN** | French  English | 66,858  105,889 | 1,379  2,209 | 4,547  6,422 | 192,191  278,590 | 528,665  576,543 |

**Baseline methods.** We compare our model against five state-of-the-art entity alignment methods:

1. RDGCN: is a deep embedding-based technique that considers attentive interaction between each KG and its dual relation by forwarding them through a two-layers GCN with highway gates. The embeddings are then compared directly to obtain the alignment result [30].
2. KGM (a.k.a KG-matching) is a deep embedding-based technique that learns the entity embeddings by employing a two-layer GCN, in which the first layer focus on local matching while the second layer learn concentrate on the global matching information [33].
3. GCNA (a.k.a GCN-align) is a deep embedding-based technique that employs GCN to produce structural and attribute embeddings, which capture the relational and attribute information of the entities, both type of embeddings are then used to discover anchor links via a set of pre-aligned entities [29].
4. BootEA is a shallow embedding-based technique that first adopts TransE model [1] to generate entity and relation embeddings. The model then reconciles the embeddings into a joint semantic space using a set of seed alignments, following by an alignment editing step to reduce error accumulation during iterations [24].
5. JAPE is a shallow embedding-based technique that generates structure embedding (using TransE model) and attribute embedding (using skip-gram model). The two types of embeddings are then used simultaneously to compute similarity score between entities [25].

It is worth noting that all the other baselines require supervision data in the form of pre-aligned entities. To respect their original settings (even though our model is handicapping), we use 30% of ground-truth as training data for the baseline methods.

Table 7: End-to-end effectiveness

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dataset** | **Metric** | **GAlign** | **RDGCN** | **KGM** | **GCNA** | **BootEA** | **JAPE** | **MuGNN** |
| **AH-EN** | Success@1  Success@10  MAP  AUC | **0.8625**  **0.9462**  **0.8931**  **0.9977** | 0.7029  0.8457  0.7250  0.9930 | 0.6519  0.7668  0.6938  0.8273 | 0.4057  0.7596  0.5270  0.9890 | 0.6019  0.8397  0.6830  0.9950 | 0.4088  0.7327  0.5210  0.9930 | 0.4779  0.8425  0.6000  0.9833 |
| **JA-EN** | Success@1  Success@10  MAP  AUC | **0.8663**  **0.9519**  **0.8987**  **0.9983** | 0.7630  0.8954  0.8110  0.9940 | 0.7009  0.8038  0.7382  0.8747 | 0.4072  0.7446  0.5270  0.9890 | 0.5867  0.8324  0.6700  0.9950 | 0.3609  0.6850  0.4710  0.9910 | 0.4866  0.8570  0.6103  0.9872 |
| **FR-EN** | Success@1  Success@10  MAP  AUC | **0.9395**  **0.9889**  **0.9582**  **0.9999** | 0.8775  0.9551  0.9060  0.9980 | 0.8782  0.9290  0.8977  0.9598 | 0.3910  0.7965  0.5270  0.9890 | 0.6203  0.8583  0.7010  0.9970 | 0.3134  0.6660  0.4320  0.9900 | 0.4896  0.8689  0.6171  0.9893 |

Table 7 answer (RQ1) by showing the end-to-end comparison of our alignment model against baseline methods on the real-world datasets.

In overall, alignment methods perform on *FR-EN* dataset better than on *ZH-EN* and *JP-EN.* This is because French is more resembled to English compare to Japanese and Chinese, thus get more precise translation. RDGCN, a deep embedding-based technique, is the best method among the baselines. GCNA and KGM also shows promising result, which prove the power of graph neural networks for entity alignment. Anyway, our technique outperforms the baselines in all scenarios while avoiding any supervision data.

Shallow embedding-based methods achieve lower accuracy. JAPE shows poorest performance on average, as the technique fail to cope with the high-level of attribute noises in the datasets. Similarity, BootEA ill-performs due to noises caused by the reconciliation step to unify the embedding spaces of different embedding types.

# CONCLUSION

## Conclusion

The thesis gives a clear definition of graph alignment problem. We discuss several research challenges which may be a hard nut to crack. Current research approaches had been investigated in detail. On top of that, we propose a novel framework of fully unsupervised network alignment for attributed networks. It is built on top of a multi-order embedding model that leverages the properties of GCN to guarantee consistency constraints. Especially, we propose an alignment refinement to detect potential noises, adjust the embedding accordingly, and make the alignment output robust to structural differences and attribute mismatches. The experiments show the superiority of our model, especially in *Success@1*, which are crucial for high-quality applications. Moreover, in term of applications, we conduct one more special experiment on an entirely different problem. But by modelling this application by graph matching task, we finally can apply our proposed model to effectively solve the problem which returns state-of-the-art performance.

## Future Direction

Although our model performs very well in graph alignment. It is true that, as our model is based on Graph Neural Network. The need of prior-node attribute is crucial. In fact, general graph does not hold this information before hand, so there is a need to define a more applicable model which can work even if the graph lacks prior node feature. Moreover, we already know that, each representation learning method may capture different aspect of the graph. Therefore, another potential direction may be finding a way to take advantage of different representation learning techniques in solving the network alignment problem.

REFERENCES

|  |  |
| --- | --- |
| [1] | H. T. S. Zhang, "FINAL: Fast attributed network alignment," in *KDD*, 2016. |
| [2] | D. S. Huan Yang, "Image Captioning with Relational Knowledge," in *PRICAI*, 2018. |
| [3] | J. X. Somaye Hashemifar, "HubAlign: an accurate and efficient method for global alignment of protein–protein interaction networks," in *Bioinformatic*, 2014. |
| [4] | H. S. T. Man, "Predict Anchor Links across Social Networks via an Embedding Approach," in *IJCAI*, 2016. |
| [5] | H. S. H. Heimann, "REGAL: Representaion Learning-based Graph Alignment," in *CIKM*, 2018. |
| [6] | B. Viswanath, "On the evolution of user interaction in facebook," in *WOSN*, 2009. |
| [7] | P. M. Giannis Nikolentzos, "Matching node embeddings for graph similarity," in *AAAI*, 2017. |
| [8] | L. Hu, "Personalized recommendation via Cross-Domain Triadic Factorization," in *WWW*, 2013. |
| [9] | Y. Dong, "Link Prediction and Recommendation across Heterogeneous Social Networks," in *ICDM*, 2012. |
| [10] | N. Q. V. Hung, "Minimizing efforts in validating crowd," in *SIGMOD*, 2015. |
| [11] | W. Z. Lu Liu, "Detecting outlier pairs in complex network based on link structure and semantic relationship," in *ESWA*, 2017. |
| [12] | Y. Xiang, "Online multi-layer dictionary pair learning for visual classification," in *ESWA*, 2018. |
| [13] | P. S. Y. Jiawei Zhang, "Meta-path based multinetwork collective link prediction," in *KDD*, 2014. |
| [14] | J. S. Zhengyu Deng, "Personalized video recommendation based on cross-platform user modeling," in *ICME*, 2013. |
| [15] | L. Liu, "Modelling of information diffusion on social networks with application to WeChat," in *Physica A: Statistical Mechanics and its Applications*, 2018. |
| [16] | J. X. Rohit Singh, "Global alignment of multiple protein interaction networks with application to functional orthology detection," in *Proceedings of the National Academy of Sciences*, 2008. |
| [17] | R. Sharan, "Conserved patterns of protein interaction in multiple species," in *Proceedings of the National Academy of Sciences*. |
| [18] | L. R. Matthews, "Identification of potential interaction networks using sequence-based searches for conserved protein-protein interactions or "interologs"," in *Genome research*, 2001. |
| [19] | T. I. Roded Sharan, "Topographic graph matching for shift estimation," in *IAPR*, 2001. |
| [20] | C. E. S. Maido Remm, "Automatic clustering of orthologs and in-paralogs from pairwise species comparisons," in *Journal of molecular biology*, 2001. |
| [21] | Q. Song, "Mining summaries for knowledge graph search," in *TKDE*, 2018. |
| [22] | Q. Wang, "Knowledge graph embedding: A survey of approaches and application," in *AAAI*, 2017. |
| [23] | F. Zang, "Collaborative knowledge base embedding for recommender systems," in *KDD*, 2016. |
| [24] | C. Wang, "Text classification with heterogeneous information network kernels," in *AAAI*, 2016. |
| [25] | Y. Zhang, "Variational reasoning for question answearing with knowledge graph," in *AAAI*, 2018. |
| [26] | C. Wang, "Text classification with heterogeneous information network kernels," in *AAAI*, 2016. |
| [27] | Z. Wang, "Cross-lingual knowledge graph alignment via graph convolutional networks," in *EMNLP*, 2018. |
| [28] | M. Chen, "Multilingual knowledge graph embeddings for cross-lingual knowledge alignment," in *arXiv preprint arXiv:1611.03954*, 2016. |
| [29] | S. Hu, "Answering natural language questions by subgraph matching over knowledge graphs," in *NIPS*, 2017. |
| [30] | Z. Sun, "Cross-Lingual Entity Alignment with Knowledge Graph Embedding," in *IJCAI*, 2018. |
| [31] | H. Zhu, "Online subgraph skyline analysis over knowledge edge graphs," in *TKDE*, 2016. |
| [32] | Z. Sun, "Bootstrapping Entity Alignment with Knowledge Graph Embedding," in *IJCAI*, 2018. |
| [33] | Y. Wu, "Relation-Aware Entity Alignment for Heterogeneous Knowledge Graph," in *arXiv preprint arXiv:1908.08210*, 2019. |
| [34] | Y. Cao, "Multi-channel graph neural network for entity alignment," in *arXiv preprint arXiv:1908.09898*, 2019. |
| [35] | K. Xu, "Cross-lingual Knowledge Graph Alignment via Graph Matching Neural Network," in *arXiv preprint arXiv:1905.11605* , 2019. |
| [36] | Q. Zhu, "Neighborhood-aware attentional representation for multilingual knowledge graphs," in *IJCAI*, 2019. |
| [37] | Z. Ying, "Hierarchical graph representation learning with differentiable pooling," in *NIPS*, 2018. |
| [38] | M. G. M. Bayati, "Algorithms for large, sparse network alignment problems," in *ICDM*, 2009. |
| [39] | J. X. R. Singh, "Global alignment of multiple protein interaction networks with application to functional orthology detection," in *PNAS*, 2008. |
| [40] | S. W. K. Shu, "User Identity Linkage Across Online Social Networks: A review," in *KDD EN*, 2017. |
| [41] | M. W. N. T. Tam, "From anomaly detection to rumour detection using data streams of social platforms," in *PVLDB*, 2019. |
| [42] | C. C. A. Y. Ren, "Meta diagram based active social networks alignment," in *ICDE*, 2019. |
| [43] | N. V. H. Nassar, "Low rank spectral network alignment," in *WWW*, 2018. |
| [44] | N. T. T. H. T. Trung, "A comparative study on network alignment techniques," in *ESWA*, 2020. |
| [45] | T. V. V. Huynh Thanh Trung, "Network Alignment by Representation on Structure and Attribute," in *PRICAI*, 2019. |
| [46] | T. V. V. Huynh Thanh Trung, "Adaptive Network Alignment with Unsupervised and Multi-order Convolutional Networks," in *ICDE*, 2020. |
| [47] | M. A. Ahmad, "Link prediction across multiple social networks," in *ICDMW*, 2010. |
| [48] | X. Q. Deng Cai, "BigBrain: An Ultrahigh-Resolution 3D Human Brain Model," in *AAAI*, 2014. |
| [49] | C. Donnat, "Learning structural node embeddings via diffusion wavelets," in *KDD*, 2018. |
| [50] | A. Coucheiro-Limeres, "Resource2Vec: Linked Data distributed representations for term discovery in automatic speech recognition," in *ESWA*, 2018. |
| [51] | J. L. Aditya Grover, "node2vec: Scalable feature learning for networks," in *KDD*, 2016. |
| [52] | S. M. G. Kollias, "Network Similarity Decomposition (NSD): A Fast and Scalable Approach to Network Alignment," in *TKDE*, 2012. |
| [53] | C. T. D. T. T. Huynh, "Network Alignment by Representation Learning on Structural and Attribute," in *PRICAI*, 2019. |
| [54] | J. A. K. Kim, "Effect of homophily on network formation," in *Comm. Nonlinear Sci. Numer*, 2017. |
| [55] | H. T. D. Koutra, "Big-Align: Fast bipartite graph alignment," in *ICDM*, 2013. |
| [56] | Q. L. Z. Wang, "Cross-lingual knowledge graph alignment via graph convolutional networks," in *EMNLP*, 2018. |
| [57] | Z. D. Q. Xie, "Unsupervised data augmentation," in *arXiv preprint arXiv:1904.12848*, 2019. |
| [58] | H. Y. C. T. Duong, "Parallel Computation of Graph embeddings," in *arXiv preprint arXiv:1909.02977*, 2019. |
| [59] | W. K. C. L. Liu, "Aligning Users across Social Networks Using Network Embedding," in *IJCAI*, 2016. |
| [60] | L. L. F. Zhou, "DeepLink: A Deep Learning Approach for User Identity Linkage," in *INFOCOM*, 2018. |
| [61] | C. L. K. Amunts, "BigBrain: an ultrahigh-resolution 3D," in *Science*, 2013. |
| [62] | N. A. R. Rossi, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2005. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1. <https://www.kaggle.com/ayushkalla1/rotten-tomatoes-movie-database> [↑](#footnote-ref-1)
2. <https://www.kaggle.com/jyoti1706/imdbmoviesdataset> [↑](#footnote-ref-2)