Network Alignment using Adversarial Graph Contrastive Learning

A B. Tech Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Aditi Goel
(1901CS04)

under the guidance of

Dr. Joydeep Chandra

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY PATNA
PATNA - 800013, BIHAR

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**Network Alignment using Adversarial Graph Contrastive Learning**" is a bonafide work of **Aditi Goel** (**Roll No. 1901CS04**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Patna under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Dr. Joydeep Chandra**

Associate Professor,

May, 2023

Department of Computer Science & Engineering,

Patna.

Indian Institute of Technology Patna, Bihar.

Acknowledgements

I am deeply grateful to my supervisor, Dr. Joydeep Chandra, for his guidance and support during my B.Tech project, and to my PhD mentor, Shruti Saxena, for her technical expertise and inspiration.

I also thank my family and friends for their unwavering support and motivation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning techniques for graph data have proven to be highly effective in addressing a wide range of problems across various fields, including biological science, social network analysis, and linguistics. In particular, the study of real-world networks has garnered considerable interest, as it enables the mining and gathering of valuable insights.

While analyzing a single network is crucial for many applications, such as link prediction, community detection, and user modelling, certain problems, such as graph clustering and alignment, requires us to address the interactions between graphs. Network alignment, which involves mapping the same entities across different networks, has numerous applications in diverse areas such as social networks, biological networks, and computer vision.

For instance, in biological science, network alignment can help identify common genes and proteins across different organisms, facilitating the identification of novel drug targets and drug side effects. In social network analysis, network alignment can be used to detect similar individuals across different social networks, helping improve targeted advertising and recommendation systems. In computer vision, network alignment can aid in object recognition and tracking by aligning different object networks.

Therefore, the study of graph-based machine learning and network alignment is an active and promising area of research with wide-ranging applications.

1.1 Objective of the Thesis

Attributed Graph

Formally, an attributed graph can be represented as $G = (V, A, X)$, where V is a set of nodes. $A \in \{0,1\}^{n \times n}$ is an adjacency matrix representing edges/connections in a graph; i.e. $A(u, v) = 0$ if there is no edge from u to v, and $A(u, v) = 1$ otherwise. $X \in \mathbb{R}^{n \times m}$ is a node attribute matrix, assuming each node has an m-dimensional attribute and n number of nodes

Problem Definition

Given two graphs $G_s = (V_s, A_s, X_s)$ and $G_t = (V_t, A_t, \mathrm{X_t})$, the network alignment problem is to calculate an alignment/similarity matrix S where $S(u, v)$ represents the matching degree between a node $u \in V_\mathrm{s}$ *and* $v \in V_\mathrm{t}$

1.2 Contrastive Learning

Self-supervised learning is a type of machine learning where a model is trained to extract useful representations from raw data without requiring explicit supervision or labels. The goal of self-supervised learning is to learn the underlying structure of the data and capture important features that can be useful for downstream tasks. This is particularly useful when labeled data is scarce or expensive to obtain, as it allows us to leverage large amounts of unlabeled data.

Self-supervised contrastive learning is a specific type of self-supervised learning that aims to learn representations by contrasting positive and negative examples in the data. In the case of graph data, positive examples might be different views of the same graph,

while negative examples might be views of different graphs. By contrasting positive and negative examples, the model learns to pull together representations of similar samples and push apart representations of dissimilar samples.[4]

To achieve this, a model encoder is trained to encode the input data into a low-dimensional representation. The goal is to train the encoder such that for any given input sample, the similarity between the representations of positive examples is much higher than the similarity between the representations of negative examples. Graph data augmentations can be used to generate different views of the same graph, allowing for self-supervision while generating graph representations. These augmentations can include adding or removing nodes and edges from the original graph.

Overall, self-supervised contrastive learning is a powerful approach to learn useful representations from large amounts of unlabeled data, which can then be used for downstream tasks such as node classification and link prediction.

1.3 The Mutual Information Maximization Principle

The InfoMax concept, which recommends learning an encoder f that maximises the mutual information or the correspondence between the graph and its representation, is the foundation upon which GCL is based. The foundation of GCL is the idea that a graph representation *f(G)* should include the characteristics of the graph G in order to differentiate it from other graphs. The GCL's main goal is as follows:

$$InfoMax : \max_{f} I(G; f(G)) \tag{1.1}$$

where *I(Y1; Y2)* represents the mutual information between two random variables *Y1* and *Y2*

1.4 Graph Information Bottleneck (GIB)

Graph representation learning has recently been explored using the information bottleneck method.

$$GIB : \max_{f} I(f(G); Y) - \beta I(G; f(G)), \tag{1.2}$$

While both the InfoMax and GIB approaches aim to optimize graph representation learning, there is a fundamental difference between the two, as can be seen by comparing equations (1.1) and (1.2). InfoMax aims to maximize information from the original network, whereas Graph Information Bottleneck minimizes this information while maximizing the information relevant with respect to downstream tasks. By removing redundant information, Graph Information Bottleneck can avoid problems encountered in some scenarios and make GNNs trained with Graph Information Bottleneck more robust against adversarial attacks and highly transferable. However, Graph Information Bottleneck requires knowledge of downstream task class labels and cannot be applied to self-supervised GNN training with limited or no labels. The challenge, therefore, is to develop methods for self-supervised learning of GNNs that are both robust and transferable.

Chapter 2

Review of Prior Works

Network alignment is the process of matching nodes across two or more networks. It involves two stages:

1. The first stage is to create a low-dimensional vector space representation of each network's nodes. This is called network embedding. Network embedding approaches generate these vector representations by capturing important structural and relational information of each node in the network.

2. The second stage is to use these vector representations to match nodes across the networks. This involves using a similarity measure to determine how similar two nodes are in different networks. Different existing approaches to network alignment differ depending on whether they focus on network embedding techniques or on using information to map nodes based on a similarity measure.

This section provides an overview of commonly used network embedding techniques, which create the vector representations, followed by an explanation of various end-to-end network alignment methods, which use these representations to match nodes across networks.

2.1 Generating Graph Embedding

In the field of network analysis, network embedding is a technique that seeks to transform the nodes of a network into a lower-dimensional vector space while preserving their relationships and properties within the network. The goal is to create an efficient, compact, and informative representation of the network that can be used for various tasks such as link prediction, node classification, and network alignment.

The network alignment problem involves aligning two or more networks with each other by mapping their nodes based on their structural and semantic similarities. Network embedding is a widely used approach to solving this problem, where the nodes of each network are first mapped to a low-dimensional vector space and then compared based on their distance or similarity in this space.

There are two main approaches to network embedding: spectral methods and neural embedding methods. Spectral methods use the spectral properties of the adjacency or Laplacian matrix of the networks to derive a low-dimensional vector space. This involves performing matrix decomposition techniques such as Singular Value Decomposition (SVD) and Eigen Decomposition, which extract the underlying structure and patterns of the network. These methods have been widely studied and developed by researchers such as Belkin & Niyogi, Tenenbaum, De Silva, & Langford, and Yan, Huang, & Jordan.

On the other hand, neural embedding methods use neural networks to learn the embedding directly from the network data. This involves training a neural network to predict the connectivity of nodes in the network or to reconstruct the network structure from the embeddings. These methods have gained popularity in recent years due to their ability to capture complex nonlinear relationships and handle large-scale networks.

2.2 Graph Alignment

Network alignment methods can be divided into two categories: supervised and unsupervised. Supervised methods rely on labelled information, such as anchor nodes, to guide the alignment process. However, these methods are limited by the availability of labelled data. On the other hand, unsupervised methods integrate complete network information using network structures, node attributes, and edge attributes, but they are computationally expensive and not scalable. Recent unsupervised methods use deep learning-based node embedding approaches like adversarial-based training and graph convolutional networks, but these approaches struggle with noisy data and a wider range of network properties.

The aim of current research is to improve unsupervised techniques to handle noisy data and a wider range of structural and attribute properties in networks.

Chapter 3

Proposed Solution

This section provides a detailed explanation of the alignment architecture of our proposed model. We start with an overview of the model and then proceed to discuss each component in detail.

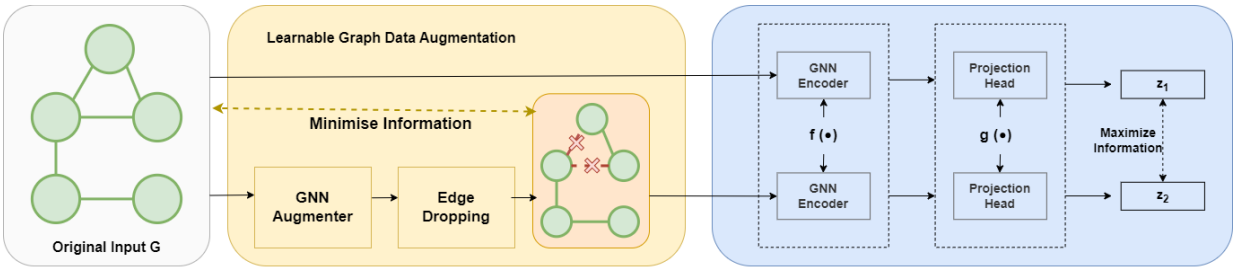

Figure 1: Architecture for learning encoder *f(.)* to get node embeddings and its instantiation. The architecture is based on a learnable edge-dropping augmentation and consists of two main components for encoding and augmentation of graph data. The GNN encoder function *f(.)* is designed to maximize the mutual information between the original graph G and the augmented graph *t(G)*. This is done while the GNN augmenter component optimizes the augmentation $T(\cdot)$ by removing information from the original graph. The instantiation proposed in this work involves using edge-dropping, where an edge e of G is randomly dropped based on the value of w_e. The value of w_e is parameterized by the GNN augmented.

Fig. 3.1 Embedding Architecture

The first component of the system utilizes the InfoMax principle with the objective of maximizing the correspondence of mutual information between the representations of the

original graph and its augmented version. The second component of the system aims to optimize the augmentation strategy with the goal of minimizing the redundant information extracted from the original graph as much as possible.

From this, we get the encoder *f(.)*, which is used to get the graph embedding z_{G_s} and z_{G_t} for the source and target graph respectively. These embeddings are then used to get the network alignment/similarity matrix S.

3.1 Learnable Edge Dropping GDA model $T_\phi(\cdot)$

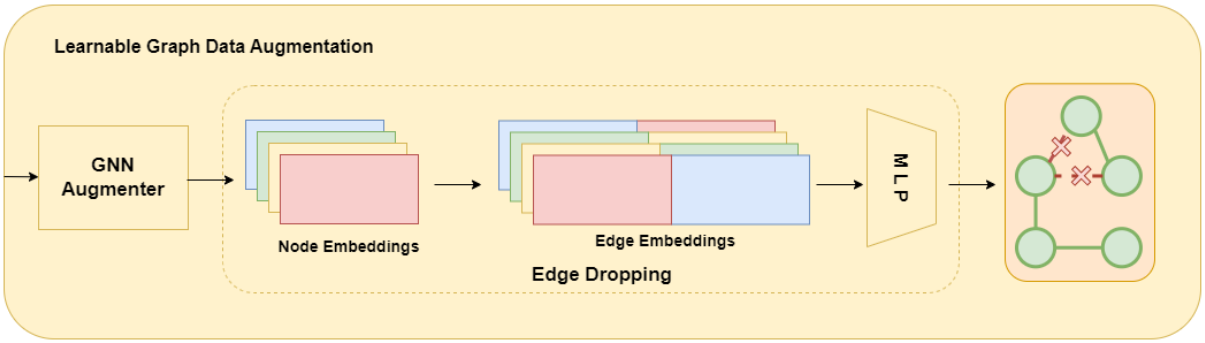

Figure 2: Architecture for learning encoder $f(\cdot)$ to get node embeddings and its instantiation based on learnable edge-dropping augmentation. It contains two components for The learnable graph augmentation is instantiated by a Graph Neural Network (GNN), which provides node embeddings derived from the original graph G. Following this, edge embeddings are constructed by concatenating two corresponding node embeddings and are then passed through a Multi-Layer Perceptron (MLP) to derive parameters for each edge w_e. These parameters indicate the probabilty of dropping each edge in the augmented graph.

Fig. 3.2 Learnable Edge Dropping GDA model $T_\phi(\cdot)$

The act of removing certain edges within a graph is commonly referred to as *edge dropping*. In this study, we have employed the edge dropping technique as a proof of concept to establish the Graph Data Augmentation (GDA) family τ. Additional forms of GDAs, including node dropping, edge adding, and feature masking, may also be combined with our approach. The basis for using edge dropping lies in the notion that effective GDAs should retain relevant information pertaining to downstream tasks. For instance, certain GRL downstream tasks such as molecule classification rely solely on the structural

fingerprints that can be expressed as subgraphs within the original graph. By selectively removing a few edges, the structural integrity of these subgraphs is not significantly altered, thereby preserving the information necessary for downstream classification.

It is important to note that this rationale does not imply the use of domain knowledge in GDA design, as the τ family is extensive and the specific GDA in question must still be optimized. Furthermore, our experimental results indicate that our instantiation exhibits outstanding performance in social network classification and molecule property regression, even in cases where the subgraph fingerprints are no longer discernible.[5]

3.2 Parameterizing $T_\phi(\cdot)$

We use a random graph model $T_\phi(G)$, where $G = (V, E)$ and T $\in \tau$, to generate a graph for each G. The generated graph, denoted as t(G), has the same set of nodes as G, but its edge set is a subset of E. Each edge in E is associated with a parameter w_e, representing its probability of being dropped. To parameterize the weights we, we use an augmenter, which is another GNN, to run on G for K layers and obtain the final-layer node representations $\{h^{(K)} \mid v \in V$ and set:

$$w_e = MLP([h^{(\mathrm{K})}{}_{\mathrm{u}}; h^{(\mathrm{K})}{}_{\mathrm{z}}]),\ where\ e = (u, z) \tag{3.1}$$

$$\{h^{(\mathrm{K})}{}_{\mathrm{v}} \mid v \in V\} = GNN - augmenter(G) \tag{3.2}$$

3.3 Regularizing $T_\phi(\cdot)$

To ensure that a generalized data augmentation (GDA) is effective in performing downstream tasks, it is important for it to retain a certain level of information. As a result, we anticipate that GDAs in the "edge dropping" family (represented by the symbol τ will not make excessively aggressive perturbations. To enforce this, we impose a constraint on

the ratio of edges dropped per graph, and regularize it by adding the term $\sum_{e \in E} w_{\mathrm{e}} / \mid E \mid$ to the objective function. Here, w_e denotes the probability of edge e being dropped. By incorporating all these factors, we arrive at the final objective function:

$$\min_{\phi} \max_{\theta} I(f_\theta(t(G)); f_\theta(G)) + \lambda_{reg} \mathbb{E}_G[\sum_{e \in E} \frac{w_{\mathrm{e}}}{\mid E \mid}] \tag{3.3}$$

$$where\ t(G) \sim T_\phi(G);\ G \sim P_G,$$

It is important to note that in this context, the symbol ϕ pertains to the set of learnable parameters of the augmenter GNN and MLP, which are utilized to derive the probabilities represented by w_e. On the other hand, the symbol θ corresponds to the learnable parameters of the GNN *f*.

3.4 Objective Estimation Equation (3.3)

In our particular implementation, the estimation of the second term, which pertains to regularization, is relatively straightforward and can be accomplished through empirical means. Conversely, for the first term that concerns mutual information, we utilize InfoNCE as the estimator. This method is renowned for being a lower bound of mutual information and is commonly used for contrastive learning. During training, a batch of m graphs $\{G_i\}_{i=1}^m$ is provided, and mutual information for the minibatch is estimated by calculating the cosine similarity using the following approach.

$$I(f_\theta(G); f_\theta(t(G)) = \frac{1}{m} \sum_{i=1}^{m} \log \frac{exp(sim(z_{i,1}, z_{i,2}))}{\sum_{i'=1, i' \neq i}^{m} exp(sim(z_{i,1}, z_{i',2}))} \tag{3.4}$$

3.5 Network Alignment

This is the concluding element responsible for producing the alignment matrix, $S \in \mathbb{R}^{n_s \times n_t}$, which enables us to infer all pairs of anchor nodes in G_s and G_t (with nodes n_s and n_t respectively) directly. The calculation of S is carried out using the embeddings generated by the encoder, Z_G, as input.

$$S = (Z_{G_s})(Z_{G_t})^T \tag{3.5}$$

Each entry in S at *(u,v)* indicates the similarity score between the corresponding nodes. Therefore, the row with the highest value in *S(u)* is considered as the anchor for the most similar node in the target network.

Fig. 3.3 Network Alignment

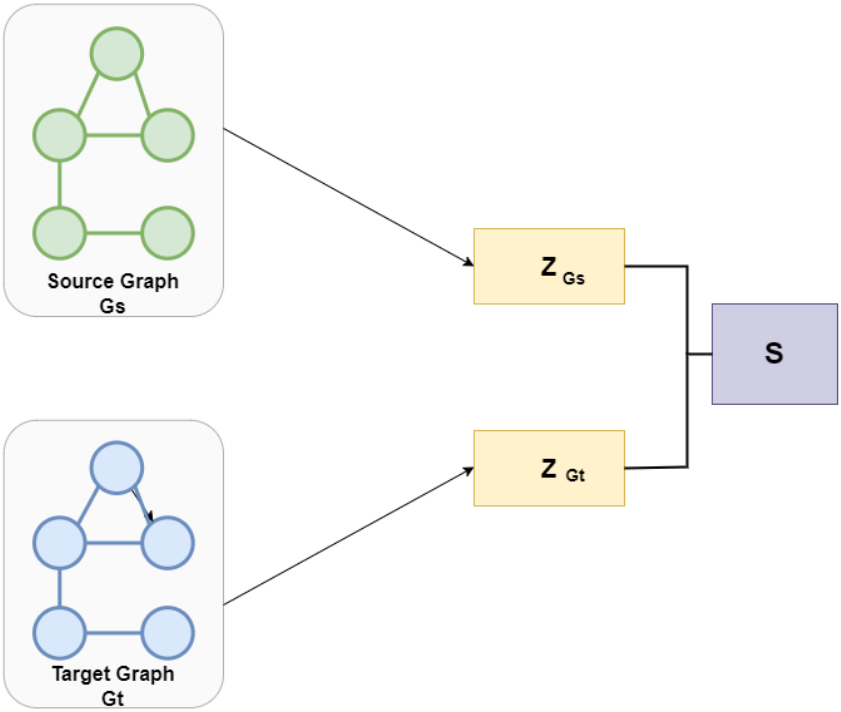

Figure 3: After getting the graph embeddings Z_{G_s} and Z_{G_t} for the source and target network respectively, we can calculate the similarity matrix S.

Chapter 4

Experimental Setup

This section is dedicated to presenting two real-world datasets that were utilized in our experiments. Subsequently, we describe the various metrics employed to assess our model and baselines in Section 4.2. Following that, we elaborate on the algorithm for training the learnable GDA, followed by the implementation details discussed in Section 4.3, and the algorithm presented in Section 4.4.

4.1 Datasets

Below, we present the details of the two real-world datasets used to evaluate the effectiveness of the proposed method.

- *Douban Online vs Douban Offline*: Douban is a social networking platform in China that has two types of networks: an Online network consisting of 3906 nodes that links user friends virtually, and an Offline network made up of 1118 nodes that is established through users' co-occurrence in social events (Du et al., 2019; Trung, Van Vinh, Tam, Yin, Weidlich, & Viet Hung, 2020). The graph representation of users in both networks includes their locations as node attributes, with edges connecting user friends in the network

- *Allmovie vs IMDB*: AllMovie and IMDB are movie guide services available online, comprising of 6011 and 5713 nodes respectively (Trung, Van Vinh, Tam, Yin, Weidlich, & Viet Hung, 2020). The nodes in the graph represent movies, and an edge connects two movie nodes if they share at least one actor. The node attributes encompass movie details such as genre and cast.

Network	**#Nodes**	**#Edges**	**#Attributes**
Douban Online	3906	8164	538
Douban Offline	1118	1511	538
Allmovie	6011	124709	14
Imdb	5713	119073	14

Table 4.1 Datasets

4.2 Baseline Models

We compare our model with the existing state-of-the-art research works which we discuss next:

1. **REGAL** (Heimann et al., 2018)[2] is an unsupervised spectral method that models the alignment matrix using low-rank matrix approximation based on topology and node feature similarity.

2. **GAlign** (Trung, Van Vinh, Tam, Yin, Weidlich, & Viet Hung, 2020)[6] is an unsupervised multi-order embedding model that leverages the properties of GCN for attributed networks. Additionally, it employs a perturbation mechanism to handle network noise and consistency violations.

3. **CENALP** (Du et al., 2019)[1] is an unsupervised method that jointly performs link prediction and user alignment tasks. It captures structural properties from separate networks using a cross-network embedding method that employs random walks.

4. **BRIGHT** (Yan et al., 2021)[7] is an unsupervised approach that learns from multiple graphs simultaneously using a cross-network embedding method employing random walks.

5. **FINAL** (Zhang & Tong, 2016)[8] is a spectral method that performs alignment on attributed graphs by extracting their structural similarity, node feature similarity, and edge feature similarity.

6. **DeNA** (Derr, Karimi, Liu, Xu, & Tang, 2021)[3] is an unsupervised approach that uses adversarial-based learning to map one graph into another and further aligns nodes efficiently using the mapped graphs.

4.3 Implementation Details

In our implementation, we have used a fixed regularization weight $\lambda_{reg} = 5$

Reproducibility Environment: Our implementation utilizes Pytorch libraries and was executed on a system equipped with an NVIDIA GeForce GTX 1080 Ti GPU with a memory of 11 GB.

4.4 Algorithm

Input: Source graph $\mathrm{G_s}$, Target graph $\mathrm{G_t}$
Encoder $f_\Theta(\cdot)$; Augmenter $T_\Phi(\cdot)$; Projection Head $g_\Psi(\cdot)$;
Hyper-Params: λ_{reg}; α, β
Output: Alignment Matrix S

begin
  for *some iterations* **do**
    for G_n *in* (G_s, G_t) **do**
      for n = *1 to* N **do**
        $h_n^1 = f_\Theta(G_n)$
        $z_n^1 = g_\Psi(h_n^1)$
        $t(G_n) \sim T_\Phi(G_n)$
        set $p_e, \forall e \in E_n$ **from** $t(G_n)$
        $\mathcal{R}_n = \sum_{e \in E_n} p_e / |E_n|$
        $h_n^2 = f_\Theta(t(G_n))$;
        $z_n^1 = g_\Psi(h_{2,n})$
      end
      define $\mathcal{L}_n = -\log \frac{\exp(sim(z_{1,n}, z_{2,n}))}{\sum_{n'=1, n' \neq n}^{N} \exp\left(sim(z_{1,n}, z_{2,n'})\right)}$
      `/* calculate NCE loss */`
      $\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}_n$
      `/* calculate regularization term */`
      $\mathcal{R} = \frac{1}{N} \sum_{n=1}^{N} \mathcal{R}_n$
      `/* update augmenter parameters using gradient ascent */`
      $\Phi \leftarrow \Phi + \alpha \nabla_\Phi (\mathcal{L} - \lambda_{\text{reg}} * \mathcal{R})$
      `/* update enocder parameters using gradient descent */`
      `/* update projection head parameters using gradient descent */`
      $\Theta \leftarrow \Theta - \beta \nabla_\Theta(\mathcal{L}); \quad \Psi \leftarrow \Psi - \beta \nabla_\Psi(\mathcal{L})$
    end
  end
  `/* obtain alignment results */`
  $S = (Z_{G_s})(Z_{G_t})^T$
  return *Alignment Matrix* S
end

Chapter 5

Results and Analysis

Our model's performance is evaluated by comparing it with several self-supervised baseline models.

5.1 Evaluation metrics

- ***Acc@1***: Top-1 Accuracy (Acc@1) is used to measure the accuracy of the proposed method in identifying the top-ranked anchor user pair across networks. It is calculated by dividing the number of accurately identified user pairs at the top of the ranked list by the total number of ground truth anchor user pairs.

$$Acc@1 = \frac{\#\{node\ pairs\ identified\ correctly\}}{\#\{ground\ truth\ anchor\ pairs\}} \tag{5.1}$$

- ***Acc@q***: Acc@q is the ratio of the number of true positive anchor matches that occur in the top-q list based on the prediction scores of a model to the total number of groundtruth anchor links. For each source node, a list $R(u_s)$ is created which comprises of the highest q values in the row $S(u_s)$. $R(u_s)$ comprises of the best potential anchor nodes from the target network.

- ***AUC***: The position of the true anchor user of u_s in the sorted list $R(u_s)$ is represented

by ra, and t denotes the number of other possible anchor users, i.e., *len(R(u_s)* - 1.

$$AUC = \frac{t + 1 - ra}{t} \tag{5.2}$$

- ***MAP***: The calculation of MAP involves computing the Average Precision (AP) for all users that need to be aligned, and is computed as

$$MAP = mean(\frac{1}{ra}) \tag{5.3}$$

5.2 Results

Datasets	Metric	FINAL	REGAL	CENALP	GAlign	DeNA	Ours
Allmovie-IMDB	Acc@1	0.765	0.095	0.486	0.775	0.728	0.816
	Acc@10	0.950	0.386	0.832	0.874	0.837	0.867
	MAP	0.846	0.188	0.5691	0.812	0.704	0.856
	AUC	0.989	0.988	0.958	0.993	0.978	0.991
Douban Online –Offline	Acc@1	0.438	0.043	0.226	0.441	0.430	0.427
	Acc@10	0.771	0.227	0.458	0.780	0.569	0.791
	MAP	0.554	0.099	0.303	0.554	0.485	0.528
	AUC	0.987	0.942	0.718	0.990	0.958	0.992

Table 5.1 Comparision of baselines with our model

5.3 Analysis

Table 5.1 presents a comprehensive comparison of our alignment model and baseline methods. Across all datasets, our model consistently outperforms all baselines in terms of key metrics such as MAP, AUC, and Acc@1. Among the baseline methods, FINAL is the best-performing and is able to compete with our model in the Allmovie-IMDB dataset in terms of Acc@q. The reason for this is that FINAL is a state-of-the-art method that, like our model, considers both structural and node attribute information. Additionally, FINAL utilizes prior alignment data as supervision, which our method does not consider.

Chapter 6

Conclusion

In our work, we have introduced a new principle for self-supervised learning in network alignment. Our proposed approach goes beyond the conventional InfoMax objective and suggests that the optimal Graph Neural Network (GNN) encoders, which are independent of downstream tasks, should capture the minimal sufficient information to identify each graph in the dataset. To achieve this goal, we propose to enhance graph contrastive learning by optimizing graph augmentations in an adversarial way. Based on this principle, we have developed a practical implementation that uses learnable edge dropping. We have thoroughly analyzed and demonstrated the advantages of our approach using real-world datasets for network alignment in self-supervised learning settings

References

[1] Alnaimy, S., and Desouki, M. Expanded graph embedding for joint network alignment and link prediction. *Journal of Big Data 9* (04 2022).

[2] Heimann, M., Shen, H., Safavi, T., and Koutra, D. REGAL. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (oct 2018), ACM.

[3] Li, Y., and Liu, H. Dena: Display name embedding method for chinese social network alignment. *Neural Comput. Appl. 35*, 10 (dec 2022), 7443–7461.

[4] Saxena, S., Chakraborty, R., and Chandra, J. Hcna: Hyperbolic contrastive learning framework for self-supervised network alignment. *Information Processing Management 59*, 5 (2022), 103021.

[5] Suresh, S., Li, P., Hao, C., and Neville, J. Adversarial graph augmentation to improve graph contrastive learning, 2021.

[6] Trung, H. T., Van Vinh, T., Tam, N. T., Yin, H., Weidlich, M., and Viet Hung, N. Q. Adaptive network alignment with unsupervised and multi-order convolutional networks. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)* (2020), pp. 85–96.

[7] Yan, Y., Zhang, S., and Tong, H. Bright: A bridging algorithm for network alignment. pp. 3907–3917.

[8] Zhang, S., and Tong, H. Final: Fast attributed network alignment. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2016), KDD '16, Association for Computing Machinery, p. 1345–1354.