EXPLORING THE POWER OF TEXT-RICH GRAPH REPRESENTATION LEARNING

BY

QI ZHU

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2023

Urbana, Illinois

ABSTRACT

During my doctoral research, I observed that many applications related to graphs cannot be captured by existing simple network models. Lots of real networks exhibit massive text information on various type of objects, also known as text-rich or text-attributed networks. Traditional graph representation learning (*e.g.* network embedding) largely overlook the complex textual information within nodes and edges. Consequently, many recent graph mining algorithms employ supervised representation learning using neural networks on graph-structured data, specifically graph neural networks (GNNs). In this dissertation, I am motivated to bridge the gap between the frontier machine learning techniques and real-world problems on graph structured data (*e.g.* web-scale retrieval and classification). Two primary obstacles have hindered previous work on text-rich graphs. First, they assume *sufficient* and *well-posed* task-specific annotations. Second, an *abundant* computation budget is required for graph neural network training and inference, which is unrealistic considering large language model with billions parameters.

From a practitioner's perspective, my research follows *label efficient* and *parameter efficient* principles to design graph representation learning algorithms for *effective* and *efficient* modeling of text-rich graphs. The first part of my dissertation work focuses on *label-efficient* representation learning, which reduces the need for extensive annotation across various tasks and graphs. Then I will introduce my recent efforts on seamless integration of language models and graph neural networks without excessive amount of training parameters. In contrast to existing work that develops powerful architectures for specific applications, my thesis overcomes the barriers to achieving *flexibility* and *efficiency* in general graph neural networks. Therefore, the proposed models can not only achieve superior performance in selected applications, but also enhance the capacity of any existing graph mining tasks.

Together with all these efforts, the developed algorithms improve the adaptation of existing graph neural networks on more sophisticated text-rich networks and seek a more powerful representation learning paradigm in this area.

To my parents, for their love and support.

ACKNOWLEDGMENTS

First, I would like to express my sincere gratitude to my Ph.D. advisor Professor Jiawei Han. One of his standout teaching styles is the constructive criticism of our research endeavors. He always encourages me to tackle real problems and never hesitate to provide support when we reach a consensus. This approach has been instrumental in my growth and development as a young researcher. Moreover, he has been a role model for me in conducting research, collaborating with others, and leading a project. I arrived UIUC with no prior research experience, and now, I return with my own research thesis. I am immensely grateful for the upcoming career path he has paved for me through your advice.

My research was supported in part by NSF, DARPA, ARL. I would like to thank my thesis committee members, Professor Hari Sundaram, Professor Hanghang Tong, and Doctor Bryan Perozzi. This journey wouldn't have been as fruitful without your valuable comments, which have been with me since my preliminary exam.

During my doctoral study, I have received numerous support and help from Professor Han's data mining group: Carl J. Yang, Chao Zhang, Yu Shi, Chanyoung Park, Xiang Ren, Haonan Wang, Yu Zhang, Bowen Jin, Yizhu Jiao, Ahmed El-Kishky, Yuning Mao, Fang Guo, Liyuan Liu, Huan Gui, Jiaming Shen, Meng Qu, Honglei Zhuang, Ming Zhong, Siru Ouyang, Suyu Ge, Shivam Agarwal, Honglei Zhuang, Shi Zhi, Xiaotao Gu, Xuan Wang, Sha Li, Di Jin, Jinfeng Xiao, Fang Guo, Xinyang Zhang, Jiaxin Huang, Yu Meng, Yanru Qu, Ellen Wu and Dongming Lei. The days and nights at the university town probably are the most unforgettable time in my 20s. I enjoy working with the best talent in the field and learning from your dedication and I will insist high standard on research throughout my career.

I am also fortunate to collaborate and get advice from other researchers inside and outside of UIUC: Bryan Perozzi, Luna Dong, Natalia Ponomareva, Da Zheng, Christos Faloutsos, Xiusi Chen and Yang Yang. I have learned a lot from their collaboration and internship mentoring. Besides, I would like to thank Professor Deng Cai and Professor Xiaofei He from Zhejiang University. They introduce me to top-tier research and are always patient with me when I have limited knowledge about the field.

Lastly, and above all, I owe my deepest gratitude to my family who have provided me with unconditional love throughout this journey: Xiaoping Gu, Jixian Zhu, Yuki, and Shuqi He. You always lift my spirits in every moment of sorrow, and this dissertation is dedicated to you. Thank you!

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Information networks are backbone of modern data systems: tons of daily social posts are circulated among users on Facebook and Twitter; a huge number of papers are published and cited in academic networks; hospitals, clinics, pharmacies, and labs are interconnected to provide comprehensive patient care, share medical records, and coordinate research. There exists a wide spectrum of network models such as heterogeneous graph, attributed graph, multi-view and hypergraph. Representation learning is a method of automatically learning appropriate features (often referred as embeddings) from raw data for a specific task. Instead of relying on hand-engineered features, which can be tedious and potentially limited in expressiveness, representation learning allows a model to learn these features from the data itself. In the realm of graph-structured data, representation learning is most common and convenient for majority of problems. For instance, one could train a multi-layer neural network classifier using node embeddings for fraud detection in financial networks. Similarly, given a product node, one could employ similarity search for product recommendations within an e-commerce network. Early graph representation learning mainly focus on the structural information such as first-order and second-order node proximity. Several years later, researchers introduced neural networks to integrate node features into representation learning, which further enabled end-to-end learning with task-specific labels. Now, Graph Neural Networks (GNNs) has become the most popular representation learning approach on attributed graph as it benefits from both graph inductive bias and training supervisions.

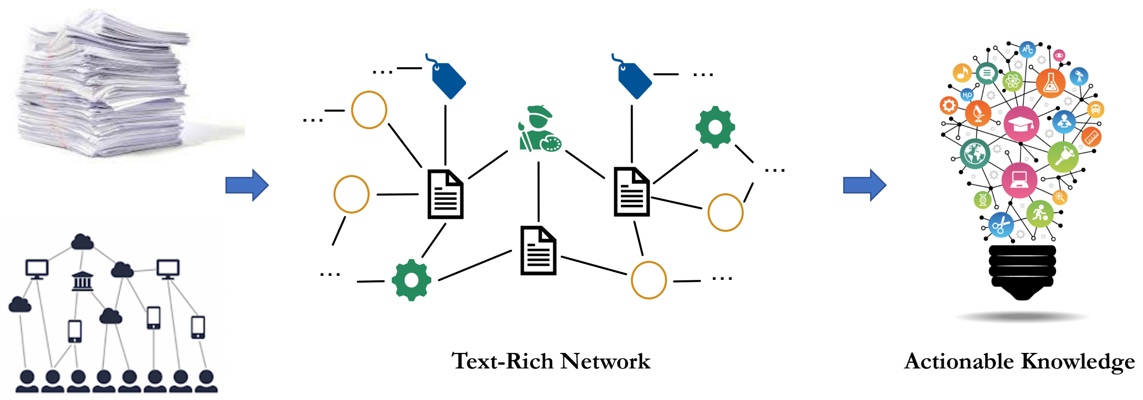


Figure 1.1: An overview of turning text-rich graphs into knowledge.

1.1 CHALLENGES OF MODELING TEXT-RICH GRAPHS

Large amount of real-world networks exhibit massive text information on various type of objects, also known as text-rich or text-attributed graphs. Existing GNNs can be applied to text-rich networks but more in-depth analysis are necessary to leverage its power. My research is motivated to examine the limitation of the current graph representation learning approaches on text-rich graphs. Furthermore, text data has emerged as the focal point among various data modalities in the era of Generative AI. Figuring out how to harness the power of language models, or foundational models, is a crucial step in advancing machine learning on text-rich graphs. In this dissertation, I summarize several unique challenges of modeling text-rich graphs into the following bullet points:

Representation learning with non IID training data. A text-rich graph is *attributed*: both attributes and context of a node determines the underlying data distribution. One presumption of most machine learning algorithms is the availability of independent and identically distributed (IID) data. However, the issue of biased training labels can become prominent, as uniform annotation accounting for both textual and structural distributions is seldom assured. Developing graph neural networks that can perform reliably with ill-posed training data is especially appealing for domains that require high-stakes decision-making or expensive annotation cost.

Transfer learning between different graphs for the same task with guarantees.

Neural networks require a significant amount of effort, from model training to deployment. Once the model is trained, developers often employ to different applications of the same task. For instance, the need for an abuse detection model is evident in various domains of the E-commerce graph. While most GNN architectures are not very complicated, the training of GNNs can still be costly regarding both memory and computation resources on real-world large-scale graphs. Moreover, it is intriguing to transfer learned structural information across different graphs and even domains in settings like few-shot learning. However, it is unclear in what situations the models will excel or fail especially when the pre-training and fine-tuning tasks are different.

Transfer learning for different types of nodes using minimal supervision. A text-rich graph is usually *heterogeneous*. The heterogeneity of real-world networks manifests in the typed nature of nodes and edges. Can we handle the same task for different types of nodes or edges? For example, we want to use the production recommendation model to obtain meaningful user representations for similarity search. One great benefit of traditional graph representation learning (*i.e.* graph embedding) is its multi-task adaptability with minimal supervision. Current research on Graph Neural Networks focuses on its predictive power on the training task without paying enough attention on its zero-shot or few-shot generalization on other types of nodes or edges.

Joint modeling of GNNs and pre-trained language models. In heterogeneous text-rich networks, the task of representation learning is more challenging due to (1) *presence or absence of text*: Some nodes are associated with rich textual information, while others are not; (2) *diversity of types*: Nodes and edges of multiple types form a heterogeneous network structure. As pretrained language models (PLMs) have demonstrated their effectiveness in obtaining widely generalizable text representations, a substantial amount of effort has been made to incorporate PLMs into representation learning on text-rich networks. How to fine-tune a graph-enhanced language model on graph structured data has been a research focus since the surge in popularity of language models.

Leverage the power of foundation models on text-rich graphs. In my last year of doctoral research, the striking result of ChatGPT reveals the superior generative and predictive power of large language models. However, existing research on jointly modeling text and graph structures either incurs high computational costs or offers limited representational power. These foundation models (*i.e.* Large Language Models) typically have 100 times more parameters than previous language models do. Graph Neural Networks are notorious for their overhead arising from aggregating neighborhood information. As a result, the actual batch size becomes dramatically larger than the original one. It is even impossible to adopt existing graph-enhanced langauge model fine-tuning algorithms on foundation models like GPT-2, Llama-2 or Falcon on a powerful A100 GPU. To the light of this, Parameter Efficient Fine-Tuning (PEFT) is a technique aimed at fine-tuning large language models in a way that requires fewer parameters and therefore, is more computationally efficient. I am fascinated by foundational models and believe in the necessity of harnessing their power for graph-structured data.

1.2 RESEARCH OVERVIEW

In my doctoral research, I investigated the aforementioned research problems and endeavored to address these issues. My dissertation outlines two suites of algorithms that focus on (1) **label-efficient** graph learning on text-rich graphs and (2) graph-aware (large) language model **parameter-efficient** fine-tuning on text-rich graphs, respectively. These two set of algorithms cover different parts of the research questions that I intend to answer in my thesis, and they can be further composed into a systematic pipeline that provides a *robust, flexible* and *efficient* solution for most of the applications on text-rich graphs.

1.2.1 Label-Efficient Graph Learning on Text-Rich Graphs.

Robust training of GNNs under distribution shift. Text-rich graphs are one of the most complicated network model proposed in the research community of graph machine learning. Thus, *robust* training and *generalizable* inference of GNNs on them should be approached with caution and handled differently than those neural networks on Euclidean or graph-structured data. Through careful consideration of distribution shift caused by biased training data, shift-robust GNNs [1, 2] are designed to account for distributional differences between biased training data and a graph's true inference distribution from two perspective. The first is to alleviate the distribution shift caused by message passing. Particularly, it can be applied to any Graph Neural Networks with parametric message passing mechanism. Second, a regularization term is introduced for GNNs that operate with a pre-computed graph inductive bias. In this way, my shift-robust training framework can work for most existing GNNs when IID training data is infeasible to obtain. We illustrate the effectiveness of SR-GNN in a variety of experiments with biased training datasets on common GNN benchmark datasets for semi-supervised learning, where we see that SR-GNN outperforms other GNN baselines in accuracy, addressing at least ∼40% of the negative effects introduced by biased training data. On the largest dataset we consider, `ogb-arxiv`, we observe a 2% absolute improvement over the baseline and are able to mitigate 30% of the negative effects from training data bias.

Transfer learning across different types of nodes. While much of the graph learning is conducted within homogeneous networks, realistic representation learning frameworks [3] need to handle heterogeneous types of nodes and edges in real world. On text-rich graphs, I believe it is important for a graph neural network that is trained on node type A can benefit other node types B,C,D. Consider the example of entity matching, CG-MuAlign [4] is proposed to jointly align multiple types of entities, collectively leverage the textual and structural information from neighborhood and hence it can generalize well to new entity types without supervision. In the movie domain, CG-MuAlign is trained on the type "Person" or "Film" and then applied to the other using a limited amount of training data. Its performance achieves an F1 score of 80% and 81%, compared to the fully supervised scores of 82% and 88%, respectively. In this study, I employ a classic collective learning algorithm drawn from the domain of relational classification. During the training of graph neural networks, CG-MuAlign enhances generalization to different node types through metric learning, even when only one type of training data is used. By enforcing the proximity of training pairs, we indirectly prompt their neighborhoods to adopt similar representations.

Transfer learning across different graphs. Researchers introduce the family of graph neural networks (GNNs) that are capable of inductive learning and generalizing to unseen nodes given meaningful node features. Yet, most existing GNNs require task-specific labels for training in a semi-supervised fashion to achieve satisfactory performance, and their usage is limited to single graphs where the downstream task is fixed. Much research has centered on designing pretext tasks for pre-training the GNNs, followed by fine-tuning these models for specific downstream tasks. However, neither a predefined transfer learning measure nor a generalization bound is provided. This omission is primarily because analyzing the data distribution of GNNs presents significant challenges [5]. To answer this question, EGI [6] (*Ego-Graph Information maximization*) establishes a theoretically grounded and practically useful framework for the transfer learning of GNNs. Firstly, it proposes a novel view towards the *essential graph information* and advocate the capturing of it as the goal of transferable GNN training, which motivates the design of EGI to analytically achieve this goal. Secondly, when node features are structure-relevant, I conduct an *analysis of* EGI *transferability* regarding the difference between the local graph Laplacians of the source and target graphs. Controlled synthetic experiments are further conducted to directly justify our theoretical conclusions. Comprehensive experiments on two real-world network datasets show consistent results in the analyzed setting of direct-transferring, while those on large-scale knowledge graphs show promising results in the more practical setting of transferring with fine-tuning.

1.2.2 Parameter Efficient Tuning of Graph-Aware Language Models.

Graph-enhanced languge model for author name disambiguation. At the model level, the combination of graph neural networks [7] and language models [8], namely, GNN-LMs has gained massive attention recently. In bibliographic network, Author name disambiguation (AND) serves as a core component of modern academic search systems to curate author profiles and bibliometrics. I found the representation power of current GNN-LMs are not fully exploited to improve the accuracy of AND. In this work, we propose a unified model – graph-enhanced language model (*i.e.*GAND) that enables (1) efficient fine-tuning with limited supervision and (2) joint modeling of the text information and relations between documents. We further develop a multi-task fine-tuning objective, which not only mitigates potential distribution shift on testing data but also showcases higher efficiency compared to existing fine-tuning objectives such as triplet loss. Experiments on two real datasets for name disambiguation demonstrate the superior performance of GAND on multiple clustering metrics.

Graph-aware parameter-efficient tuning of large language models. Previously, language models like BERT and GPT already achieved state-of-the-art performance on classical natural language processing tasks such as text classification, question answering, document summarization and *etc.* More recent large language models (LLMs) like T5, GPT-3, and Llama-2 have showcased astounding capabilities, demonstrating an unparalleled level of generalization to previously unseen tasks. It's clear that integrating these advanced language models into GNN-LMs can further enhance performance across various predictive tasks. However, harnessing LLMs for text-rich graphs is exceedingly challenging from a practical perspective. A smaller Llama 7B model has 70 times more parameters than the widely-used BERT-base language model. Even on normal text data, it is expensive to fine-tune a pre-trained Llama or GPT-3. Motivated by the literature of parameter-efficient tuning, we propose GraphAdapter to harness the power of the LLM without fine-tuning its weights on text-rich Graphs. Given a graph, an adapter GNN is trained to reduce the LLM's error in predicting the next word of text sequences on nodes. Once trained, this GNN adapter can be seamlessly fine-tuned for various downstream tasks. Through extensive node classification experiments across multiple domains, GraphAdapter demonstrates an average improvement of $\sim$5% while being more computationally efficient than baselines. We further validate its effectiveness with various language models, including RoBERTa, GPT-2, and Llama 2.

Organization. The remaining of this thesis is organized as follows.

- In Chapter 2, I provide a comprehensive literature review on text-rich graph modeling, particularly focusing on label-efficient and parameter-efficient methods discussed in my thesis.
- In Chapter 3, I present a shift-robust training framework for GNNs that is less sensitive to the quality of training data on text-rich graphs.
- In Chapter 4 and 5, I present two specific designs of graph neural networks that facilitates transfer learning across different node types and graphs, respectively.
- In Chapters 6 and 7, using either a language model or a large language model as a text encoder, I introduce a parameter-efficient training loss for LM and a parameter-efficient model architecture for LLM.
- In Chapter 8, I conclude my dissertation by summarizing my current achievements and notable future directions.

CHAPTER 2: LITERATURE REVIEW

Before we dive into specific chapters, we describe some preliminary notation and knowledge of graph neural networks that are used in all chapters. Then we survey existing studies that are related to label-efficient and parameter-efficient modeling of text-rich networks.

2.1 PRELIMINARY: GNNS

Given a graph $G = \{V, E, X\}$, the nodes V are associated with their features X ($X \in \mathbb{R}^{|V| \times F}$) and the set of edges E (*i.e.* adjacency matrix A, $A \in \mathbb{R}^{|V| \times |V|}$) which form connections between them. Graph neural networks [7] are neural networks that operate on both node features and graph structures. The core assumption of GNNs is that the structure of data (A) can provide a useful *inductive bias* for many modeling problems. GNNs output node representations Z which are used for unsupervised [9] or semi-supervised [7, 10] learning. We denote the labels for SSL as $\{y_i\}$.

The general architecture Φ of a GNN consists of K neural network layers which encode the nodes and their neighborhood information using some learnable weights θ. More specifically the output of layer k of a GNN contains a row (h_i^k) which can be used as a representation for each node i, i.e. $z_i^k = h_i^k$. Successive layers mix the node representations using graph information, for example:

$$H^k = \sigma(\tilde{A} H^{k-1} \theta^k) \tag{2.1}$$

where $\tilde{A}$ is an appropriately normalized adjacency matrix[1], σ is the activation function, and $H^0 = X$. For brevity's sake, we refer to the final latent representations Z^K simply as Z throughout the work.

2.2 GRAPH MACHINE LEARNING ON TEXT-RICH GRAPHS

2.2.1 Graph neural network architectures

Graph neural networks (GNNs) such as GCN [7], GraphSAGE [10], and GAT [11] have been widely adopted in representation learning on graphs. Since real-world objects and interactions are often multi-typed, recent studies have considered extending GNNs to het-

[1]The exact use of the adjacency matrix varies with specific GNN methods. For instance, the GCN [7] performs mean-pooling using matrix multiplication: $\tilde{A} = D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}$ where I is the identity matrix, and D is the degree matrix of $(A + I)$, $D_{ii} = \sum_j (A + I)_{ij}$.

erogeneous graphs [12]. The basic idea of heterogeneous graph neural networks (HGNNs) [13, 14, 15, 16, 17, 18] is to leverage node types, edge types, and meta-path semantics [19] in projection and aggregation. For example, HAN [14] proposes a hierarchical attention mechanism to capture both node and meta-path importance; HGT [18] proposes an architecture similar to Transformer [20] to carry out attention on edge types. For more HGNN models, one can refer to recent surveys [21, 22]. Lv et al. [23] further perform a benchmark study of 12 HGNNs and propose a simple HGNN model based on GAT. Despite the success of these models, when some types of nodes carry text information, they lack the power of handling textual signals in a contextualized way. On text-rich graphs, [24, 25] are the most recent work that jointly model text semantics and heterogeneous structure (network) signal in each Transformer layer.

2.2.2 Application

Link Prediction. Entity Alignment problem can be formulated as predicting "equivalent" links among two graphs. For large-scale graph structured data, network embeddings [26, 27, 28] tranform nodes in networks into dense vectors that helps predict missing links conveniently. Methods [29, 30, 31, 32] capture the first-order information and facilitate logical inference with explicit relation embeddings. To capture higher order information, previous work models the heterogenous types of nodes and edges with pre-defined meta-structures [33, 34, 35, 36, 37]. HEER [3], instead, applies type-specific metrics on different types of edges. HetG [38] proposes a random walk based sampling strategy and models each type of neighborhood nodes differently. GATNE [16] studies the attributed multiplex heterogenous network embedding under transductive and inductive settings. HAN [39] aggregates neighbor nodes along meta-path via hierarchical attention mechanism.

Graph Alignment. Previous efforts on graph alignment mainly span on the transductive setting, where the entity attribute is not available. Traditional network alignment methods [40, 41] focus on graphs with a small number of relation types and optimize the alignment objective based on the topology. Taking the advantage of knowledge graph representation learning [29], embedding based methods [42, 43, 44, 45] embed entities from different space along with an alignment objective on the training data. Starting with limited alignment seeds, people propose to use either bootstrapping [45] or iteratively [42] align the entities. graph neural network methods designed for entity alignment [46, 47, 48] demonstrate great performance improvement for multi-lingual KG alignment.

Entity Matching. Entity Matching (EM) techniques [49, 50, 51] find all tuple pairs between two relational tables. It is composed by two major components: (1) blocking [52] utilizes heuristics to remove obviously non-matched pairs between two tables and (2) matching [49] predicts the match score for the remaining pairs. Megellan [49] is an open-source and scalable entity matching framework that uses handcraft features and various machine learning algorithms as the matching function such as SVM, random forest, *etc.*. Later, DeepMatcher [50] computes the pairwise similarity from the attribute embeddings and adopts deep neural network as the matching function. DeepER [51] is the first to consider sequence model like LSTM to model the textual attributes automatically.

Collective Entity Resolution. Researchers have noticed the correlations between labeled entities and unlabeled neighbor entities in multi-relational database and network structured data [53, 54]. Collective entity resolution considers that the decisions made on related entities are affected by each other. An unsupervised method [55] is proposed to reduce the entity resolution problem into a graph summarization problem by clustering the co-referenced entities across different knowledge graphs into a super node. People design collective features upon human-curated entity resolution rules [56]. PARIS is an unsupervised probabilistic model for ontologies alignment. HolisticEM [57] builds a graph where each node represents similarity of a pair of entities, and propagates similarity by Personalized Random Walk to make collective decisions. We carefully consider collective decisions between sub-graphs sampled around candidate pairs and boost the performance of GNN greatly.

Author Name Disambiguation. There have been lots of efforts in the field to perform author name disambiguation in bibliographic databases. Early approaches [58, 59] define various similarity metrics between pairs of articles and apply unsupervised clustering algorithms to correspond each cluster to a real-world author. Ever since feature engineering became the most critical step towards successful disambiguation. Lots of pairwise features [60, 61, 62] are proposed to train pairwise classifiers such as co-authors, affiliations, ethnicity *etc.* People also collected multiple AND datasets (*e.g.* AMiner [63], INSPIRE [60]) and trained pairwise classifier. The notable ones are random forests [64, 65] and deep neural networks [63, 66]. Similar to GAND, [67, 68, 69] constructed a document graph with relations between documents and conducted clustering on the graph. In this work, we focus on learning the document representations for author name disambiguation. To this end, both AMiner-AND [63] and AND-GAT [69] learn a neural network encoder on word embeddings. However, none of these algorithms simultaneously model deep contextualized text embeddings and relational information.

2.3 LABEL-EFFICIENT GRAPH LEARNING ON TEXT-RICH GRAPHS

2.3.1 Distributional shift and domain adaption on graph

One of the first theoretical works on domain adaptation [70] developed a distance function between a model's performance on the source and target domains to describe how similar they are. To obtain a final model, training then happens on a reweighed combination of source and target data, where weights are a function of the domain's distance. Much additional theoretical (e.g. [71]) and practical work expanded this idea and explored models which are co-trained on both source and target data. These models seek to optimize utility while minimizing the distance between extracted features distributions on both domains; this in turn led to the field of Domain Invariant Representation Learning (DIR) [72]. DIR is commonly achieved via co-training on labeled source and (unlabeled) target data. A modification to the loss either uses an adversarial head or adds additional regularizations. More recently, various regularizations using discrepancy measures have been shown to be more stable to hyperparameters and result in better performance than adversarial heads, faster loss convergence and easier training. Maximum mean discrepancy (MMD) [73, 74] is a metric that measures difference between means of distributions in some rich Hilbert kernel space. Central moment discrepancy (CMD) [75] extends this idea and matches means and higher order moments in the original space (without the projection into the kernel). CMD has been shown to produce superior results and is less susceptible to the weight with which CMD regularization is added to the loss [74]. It is important to point out that MMD and CMD regularizations are commonly used with non-linear networks on some hidden layer (e.g. on extracted features). For linear models or non-differentiable models, prior work for domain adaptation often employed importance-reweighting instead. To find the appropriate weights, the same MMD distance was often used: in kernel mean matching (KMM) to find the appropriate weights one essentially minimizes MMD distance w.r.t the instance weights [76]. Although no existing work studies the distributional shift problem in GNNs, transfer learning of GNNs [77, 78] has explored different node-level and graph-level pre-training tasks across different graphs. Alternatively, domain adaption methods have been used to optimize a domain classifier between source and target graphs [79, 80]. In addition, distribution discrepancy minimization (MMD) has been adopted to train network embedding across domains [81]. Other regularizations for the latent state of GNNs have been proposed for domains like fairness [82]. We are the first to notice the importance of distributional shift on the same graph in a realistic setting and analyze its influence on different kinds of GNNs.

2.3.2 Pre-training on graph structured data

Representation learning on graphs has been studied for decades, with earlier spectral-based methods [83, 84, 85] theoretically grounded but hardly scaling up to graphs with over a thousand of nodes. With the emergence of neural networks, unsupervised network embedding methods based on the Skip-gram objective [86] have replenished the field [26, 27, 28, 87]. Equipped with efficient structural sampling (random walk, neighborhood, *etc.*) and negative sampling schemes, these methods are easily parallelizable and scalable to graphs with thousands to millions of nodes. However, these models are essentially transductive as they compute fully parameterized embeddings only for nodes seen during training, which are impossible to be transfered to unseen graphs.

More recently, researchers introduce the family of graph neural networks (GNNs) that are capable of inductive learning and generalizing to unseen nodes given meaningful node features [7, 10, 88]. Yet, most existing GNNs require task-specific labels for training in a semi-supervised fashion to achieve satisfactory performance [7, 10, 11, 22], and their usage is limited to single graphs where the downstream task is fixed. To this end, several unsupervised GNNs are presented, such as the auto-encoder-based ones like VGAE [89] and GNFs [90], as well as the deep-infomax-based ones like DGI [91] and InfoGraph [92]. Their potential in the transfer learning of GNN remains unclear when the node features and link structures vary across different graphs.

Although the architectures of popular GNNs such as GCN [7] may not be very complicated compared with heavy vision and language models, training a dedicated GNN for each graph can still be cumbersome [93, 94]. Moreover, as pre-training neural networks are proven to be successful in other domains [8, 95], the idea is intriguing to transfer well-trained GNNs from relevant source graphs to improve the modeling of target graphs or enable few-shot learning [80, 96, 97] when labeled data are scarce. In light of this, pioneering works have studied both generative [98] and discriminative [77, 99] GNN pre-training schemes. Though Graph Contrastive Coding [100] shares the most similar view towards graph structures as us, it utilizes contrastive learning across all graphs instead of focusing on the transfer learning between any specific pairs. On the other hand, unsupervised domain adaptive GCNs [80] study the domain adaption problem only when the source and target tasks are homogeneous.

Most previous pre-training and self-supervised GNNs lack a rigorous analysis towards their transferability and thus have unpredictable effectiveness. The only existing theoretical work on GNN transferability studies the performance of GNNs across different permutations of a single original graph [101, 102] and the tradeoff between discriminability and transferability of GNNs [103]. EGI in this thesis is the first to rigorously study the more practical setting of transferring GNNs across pairs of different source and target graphs.

2.4 GRAPH-AWARE LANGUAGE MODEL AND FOUNDATION MODEL

2.4.1 Pretrained language models

Pretrained language models (PLMs) aim to learn general language representations from large-scale corpora, which can be generalized to various downstream tasks. Early studies on PLMs mainly focus on context-free text embeddings such as word2vec [86] and GloVe [104]. Recently, motivated by the fact that the same word can have different meanings conditioned on different contexts, deep language models such as ELMo [105], BERT [8], RoBERTa [106], XLNet [107], ELECTRA [108], and GPT [109, 110] are proposed to capture the contextualized token representations. These models employ the Transformer architecture [20] to capture long-range and high-order semantic dependency and achieve significant improvement on many downstream NLP tasks [111, 112].

2.4.2 Joint modeling of LMs and GNNs

Early work on leveraging pretrained language models on graph [113, 114] directly supervise language model fine-tuning through graph-related tasks, to help language models better understand the textual information in text-attributed graphs. The language model is then combined with GNNs by freezing the language model. Their results demonstrate the benefits of fine-tuning language models on text-rich graphs. However, it has clear disadvantages that LM and GNN are optimized separately. Using language models to model graph neighbors requires huge memory and time costs. To address this, some work [112] proposed freezing the language model to reduce the computation needed for cascading. Other studies [24, 25, 115, 116] have proposed the use of cascading LM-GNNs and also encoding neighboring nodes with language models. To decrease computational costs, recent efforts have explored the joint training of LMs and GNNs through methods like knowledge distillation [114] or Expectation Maximization algorithms [117]. With the breakthrough progress made by LLMs on textual tasks [118, 119], recently many works have emerged exploring how to directly utilize LLMs to understand text-attributed graphs [120]. For example, by converting the graph to text [121, 122], or by converting it to a graph representation as part of a prompt [123]. Some works also explored using large models to enhance the textual features of text-attributed graphs [124, 125].

2.4.3 Parameter-Efficient fine-tuning large language models

The ongoing quest for improving the efficiency of large-scale language models has led to several innovative techniques that maximize performance while minimizing the number of parameters updated during fine-tuning. Here we only include the most relevant ones to this thesis, that is, BitFit [126], Adapter [127], Prefix Tuning [128] and LoRA [129]. BitFit invents a simple yet powerful concept: when fine-tuning pre-trained models, only a single bit of the gradient is used. Adapters are small neural network modules added to pre-trained models that allow for task-specific fine-tuning without adjusting the original model weights. Houlsby et al. [127] introduced the adapter approach, showcasing that with minimal additional parameters, adapters could achieve performance comparable to full-model fine-tuning. Similarly, Prefix Tuning [128] prepends trainable embeddings to the original model enables the model to adapt to new tasks with a fraction of additional parameters. LoRA [129] is the abbrevation of "Low Rank Adaptation". The key innovation of LoRA lies in decomposing the weight change matrix into two low-rank matrices, A and B. Rather than directly fine-tuning the primary parameters, LoRA emphasizes updating the parameters within the A and B matrices. These parameter-effiecient fine-tuning (PEFT) techniques significantly ease the challenges of customize state-of-the-art foundational models such as GPT-3 [110], Llama-2 [118], and Falcon [130]. I draw inspiration from these pioneering works and deeply appreciate previous efforts aimed at lowering the barriers to LLMs for individual researchers.

CHAPTER 3: SHIFT-ROBUST GRAPH NEURAL NETWORKS

3.1 BACKGROUND

The goal of graph-based semi-supervised learning (SSL) is to use relationships between data (*its graph inductive bias*), along with a small set of labeled items, to predict the labels for the rest of a dataset. Unsurprisingly, varying exactly which nodes are labeled can have a profound effect on the generalization capability of a SSL classifier. Any bias in the sampling process to select nodes for training can create distributional differences between the training set and the rest of the graph. During inference any portion of the graph can be used, so any uneven labeling for training data can cause training and test data to have different distributions. An SSL classifier may then overfit to training data irregularities, thus hurting the performance at inference time.

Recently, GNNs have emerged as a way to combine graph structure with deep neural networks. Surprisingly, most work on semi-supervised learning using GNNs for node classification [7, 10, 131] have ignored this critical problem, and even the most recently proposed GNN benchmarks [132] assume that an independent and identically distributed (IID) sample is possible for training labels.

This problem of biased training labels can be quite pronounced when GNNs are applied for semi-supervised learning in practice. It commonly happens when the size of the dataset is so large that only a subset of it can afford to be labeled – the *exact* situation where graph-based SSL is supposed to have a value proposition! While the specific source of bias can vary, we have encountered it in many different settings. For example, sometimes fixed heuristics are used to select a subset of data (which shares some characteristics) for labeling. Other times, human analysts individually choose data items for labeling, using complex domain knowledge. However, even this can be rooted in shared characteristics of data. In yet another scenario, a label source may have some latency, causing a temporal mismatch between the distribution of data at time of labeling and at the time of inference. In all of these cases, the core problem is that the GNN overfits to spurious regularities as the subset of labeled data could not be created in an IID manner.

One particular area where this can apply is in the spam and abuse domain, a common area of application for GNNs [133, 134, 135]. However, the labels in these problems usually come from explicit human annotations, which are both sparse (as human labeling is expensive), and also frequently biased. Since spam and abuse problems typically have very imbalanced label distributions (e.g. in many problems there are relatively few abusers – typically less than 1:100), labeling nodes IID results in discovering very few abusive labels. In this case choosing the points to request labels for in an IID manner is simply not a feasible option if one wants to have a reasonable number of data items from the rare class.

3.2 DISTRIBUTIONAL SHIFT IN GNNS

To learn an SSL classifier, a cross-entropy loss function l is commonly used,

$$\mathcal{L} = \frac{1}{M}\sum_{i=1}^{M} l(y_i, z_i), \tag{3.1}$$

where z_i is the node representation for the node i learned from a graph neural network, and M is the number of training examples. When the training and testing data come from the same domain (i.e. $\Pr_{\text{train}}(X,Y) = \Pr_{\text{test}}(X,Y)$), optimizing the cross entropy loss over the training data ensures that a classifier is well-calibrated for performing inference on the testing data.

3.2.1 Data shift as representation shift

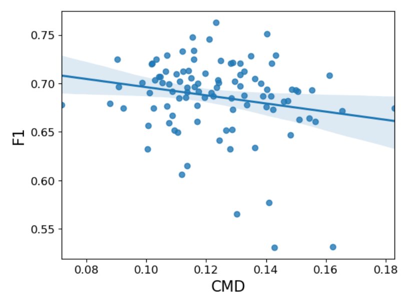
However, a mismatch between the training and testing distributions (*i.e.* $\Pr_{\text{train}}(X,Y) \neq \Pr_{\text{test}}(X,Y)$) is a common challenge for machine learning [136, 137].

In this paper, we care about the distributional shift between training and test datasets present in Z, the output of the last activated hidden layer. Given that the foundation of standard learning theory assumes $\Pr_{\text{train}}(Y|Z) = \Pr_{\text{test}}(Y|Z)$, the main cause of distribution shift is the representation shift, *i.e.* $\Pr_{\text{train}}(Z,Y) \neq \Pr_{\text{test}}(Z,Y) \rightarrow \Pr_{\text{train}}(Z) \neq \Pr_{\text{test}}(Z)$. To measure such a shift, discrepancy metrics such as MMD [138] or CMD [139] can be used. CMD measures the direct distance between distributions p and q as the following [139]:

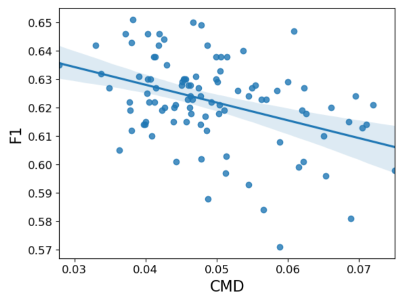
$$\text{CMD} = \frac{1}{|b-a|}\|\mathrm{E}\left(p\right) - \mathrm{E}\left(q\right)\|_2 + \sum_{k=2}^{\infty}\frac{1}{|b-a|^k}\|c_k\left(p\right) - c_k\left(q\right)\|_2, \tag{3.2}$$

where c_k is k-th order moment and a, b denotes the joint distribution support of the distributions. In practice, only a limited number of moments is usually included (e.g. k=5). In this work we focus on the use of CMD [139] as a distance metric to measure distributions discrepancy for efficiency.

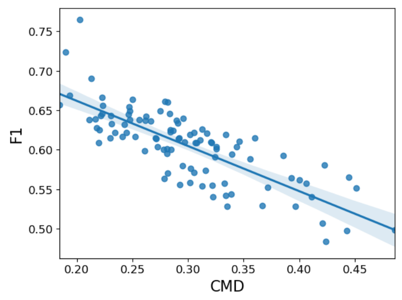
We note that GNNs (Eq (2.1)) are different from traditional neural networks, where the output for a layer K is defined as $H^k = \sigma(H^{k-1}\theta^k)$. Instead, the multiplication of the



(a) Cora



(b) Citeseer



(c) Pubmed

Figure 3.1: Distribution shift lowers performance on GNN datasets. For each dataset, we show the performance (F1:y-axis) vs their distribution shift (CMD:x-axis) for 100 biased training set samples .

normalized adjacency matrix ($H^k = \sigma(\tilde{A}H^{k-1}\theta^k)$) essentially changes the output distribution of the hidden representation via the graph's inductive bias. Hence, in a semi-supervised GNN, a biased training sample can lead to large representation shift due to both the graph's inductive bias in addition to 'normal' shift between non-IID sampled feature vectors.

Formally, we start the analysis of distributional shift as follows.

Definition 3.1 (Distribution shift in GNNs)**.** Assume node representations $Z = \{z_1, \ldots, z_n\}$ are given as an output of the last hidden layer of a graph neural network on graph G with n nodes. Given labeled data $\{(x_i, y_i)\}$ of size M, the labeled node representation $Z_l = (z_1, \ldots, z_m)$ is a subset of the nodes that are labeled, $Z_l \subset Z$. Assume Z and Z_l are drawn from two probability distributions p and q. The distribution shift in GNNs is then measured via a distance metric $d(Z, Z_l)$.

Interestingly, it can be empirically shown that the effects of distribution shift due to sample bias directly lower the performance of models. To illustrate this, we plot the distribution shift distance values (x-axis) and corresponding model accuracy (y-axis) for three common GNN benchmarks using the classic GCN model [7] in Figure 3.1. The results demonstrate that the performance of GNNs for node classification on these datasets is inversely related to the magnitude of distributional shift and motivates our investigation into distribution shift.

3.3 SHIFT-ROBUST GRAPH NEURAL NETWORKS

In this section, we will address the distributional shift problem ($\Pr_{\text{train}}(Z) \neq \Pr_{\text{test}}(Z)$) in GNNs by proposing ways to mitigate the shift for two different GNN models (Section 3.3.1 and 3.3.2, respectively). Subsequently, we introduce SR-GNN (Fig.5.2) as a general

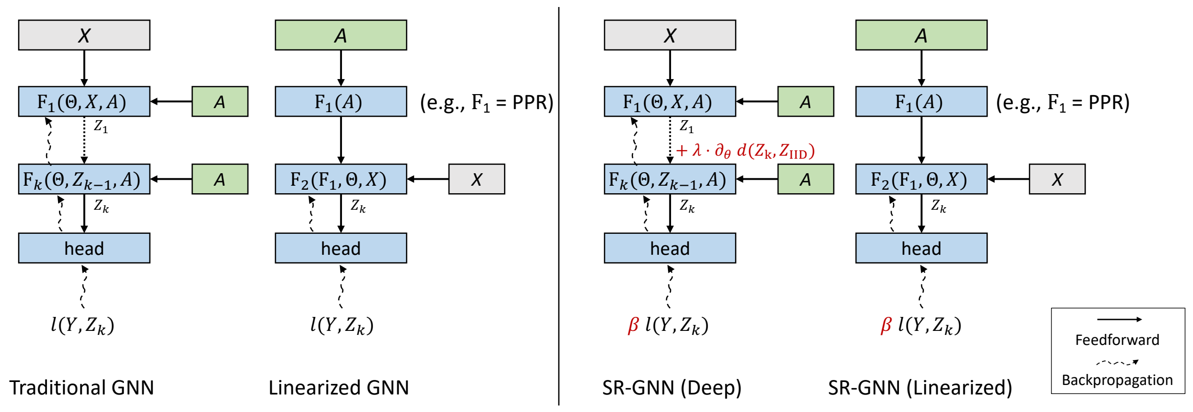


Figure 3.2: A comparison between a traditional GNN, a linearized GNN and our framework (SR-GNN).

framework that reduces distributional shifts for both differentiable and non-differentiable (e.g. graph inductive bias) sources simultaneously in Section 3.3.3.

3.3.1 Scenario 1: Traditional GNN models

We begin by considering a traditional GNN model Φ, a learnable function $\mathbf{F}$ with parameters $\boldsymbol{\Theta}$, over some adjacency matrix A:

$$\Phi = \mathbf{F}(\Theta, Z, A). \tag{3.3}$$

In the original GCN [7], the graph inductive bias is multiplicative at each layer and gradients are back propagated through all of the layers. In the last activated hidden layers, we denote a bounded node representation[2] as $Z \equiv Z_k = \Phi(\Theta, Z_{k-1}, A), Z_k \in [a,b]^n, Z_0 = X$.

Let us denote the training samples as $\{x_i\}_{i=1}^M$, the node representations are $Z_{\text{train}} = \{z_i\}_{i=1}^M$. For the test samples, we sample an unbiased IID sample from unlabeled data $X_{\text{IID}} = \{x'_i\}_{i=1}^M$ and denote the output representations as $Z_{\text{IID}} = \{z'_i\}_{i=1}^M$.

In order to mitigate the distributional shift between training and testing, we propose a regularizer $d : [a,b]^n \times [a,b]^n \to \mathbb{R}^+$ that is added to the cross entropy loss. Since Φ is fully differentiable, we can use a distributional shift metric as a regularization to directly minimize the discrepancy between a biased and unbiased IID sample like so:

$$\mathcal{L} = \frac{1}{M} \sum_i l(y_i, z_i) + \lambda \cdot d(Z_{\text{train}}, Z_{\text{IID}}). \tag{3.4}$$

[2]We use a bounded activation function here, *e.g.* tanh or sigmoid.

Here we consider the central moment discrepancy regularizer, d_{CMD}:

$$d_{\text{CMD}}(Z_{\text{train}}, Z_{\text{IID}}) = \frac{1}{b-a}\|\mathbf{E}(Z_{\text{train}}) - \mathbf{E}(Z_{\text{IID}})\| + \sum_{k=2}^{\infty} \frac{1}{|b-a|^k}\|c_k(Z_{\text{train}}) - c_k(Z_{\text{IID}})\|, \quad (3.5)$$

where $\mathbf{E}(Z) = \frac{1}{M}\sum z_i$ and $c_k(Z) = \mathbf{E}(Z - \mathbf{E}(Z))^k$ is the k-th order moment. In practice, we use moments up to the 5th order.

3.3.2 Scenario 2: Linearized GNN Models

Another family of recently proposed models for GNNs uses two distinct different functions – one for non-linear feature transformation, and another for a linear graph spreading stage,

$$\Phi = \mathbf{F_2}(\underbrace{\mathbf{F_1}(\mathbf{A})}_{\text{linear function}}, \Theta, X). \quad (3.6)$$

In such a linearized GNN model, the graph inductive bias is combined with node features by a linear function $\mathbf{F_1}$, which is decoupled from multi-layer neural network feature encoder $\mathbf{F_2}$. SimpleGCN [140] is an example of linearized model when $\mathbf{F_1}(A) = A^k X$. Another branch of linearized models [141, 142, 143] employs personalized pagerank to pre-compute the information diffusion in a graph (*i.e.* $\mathbf{F_1}(A) = \alpha(I - (1-\alpha)\tilde{A})^{-1}$) and apply it on encoded node features $F(\Theta, X)$.

In both models, the graph inductive bias is provided as an input feature to a linear $\mathbf{F_1}$. Unfortunately, as there are no learnable layers at this stage in these models, one can not simply apply the distributional regularizer proposed in the previous section. In this case, we can view training and testing samples as row-wise samples h_i from $\mathbf{F}_1(A)$. The problem of distribution shift $\Pr_{\text{train}}(Z) \neq \Pr_{\text{test}}(Z)$ can then be transformed into matching the training and testing graph inductive bias feature space $h_i \in \mathbb{R}^n$ (where n is the number of the nodes in the graph). Then to generalize from training data to testing, we can adopt an instance reweighting scheme to correct the bias, such that biased training sample $\{h_i\}_{i=1}^M$ will be similar to an IID sample $\{h'_i\}_{i=1}^M$. The resulting cross entropy loss is then

$$\mathcal{L} = \frac{1}{M}\beta_i l(y_i, \Phi(h_i)), \quad (3.7)$$

where β_i be the weight for each training instance, and l is the cross-entropy loss. We can then compute the optimal β via kernel mean matching (KMM) [144] by solving a quadratic problem,

$$\min_{\beta_i} \|\frac{1}{M}\sum_{i=1}^{M}\beta_i\psi(h_i) - \frac{1}{M'}\sum_{i=1}^{M'}\psi(h'_i)\|^2, \textbf{ s.t. } B_l \leq \beta < B_u \tag{3.8}$$

It tries to match the mean elements in a kernel space $k(\cdot,\cdot)$ on the domain $\mathbb{R}^n \times \mathbb{R}^n$. Specifically, $\psi : \mathbb{R}^n \to \mathcal{H}$ denotes the feature map to the reproducing kernel Hilbert space(RKHS) introduced by kernel k. In our experiment, we use a mixture of gaussian kernel $k(x,y) = \sum_{\alpha_i} \exp(\alpha_i\|x-y\|_2), \alpha_i = 1, 0.1, 0.01$. The lower B_l and upper bound B_u constraints are there to make sure that most of the instances get some reasonable weight, as opposed to only a few instances getting non zero weight. In practice, we have multiple classes in the label space. To prevent label imbalance introduced by β, we further require that the sum of β for a specific class c remains the same before and after the correction, $\sum_i^M \beta_i \cdot \mathbb{I}(l_i = c) = \sum_i^M \mathbb{I}(l_i = c), \forall c$.

3.3.3 Shift-Robust GNN Framework

Now we propose Shift-Robust GNN (SR-GNN) - our general training objective for addressing distributional shift in GNNs:

$$\mathcal{L}_{\mathsf{SR-GNN}} = \frac{1}{M}\beta_i l(y_i, \Phi(x_i, A)) + \lambda \cdot d(Z_{\text{train}}, Z_{\text{IID}}). \tag{3.9}$$

The framework consists of both a regularization for addressing distributional shift in learnable layers (Section 4.1) and an instance reweighting component which is capable of handling situations where a graph inductive bias is added after feature encoding (Section 4.2).

We will now discuss a concrete instance of our framework, by applying it to the APPNP [141] model. The APPNP model is defined as:

$$\Phi_{\text{APPNP}} = \underbrace{\left((1-\alpha)^k\tilde{A}^k + \alpha\sum_{i=0}^{k-1}(1-\alpha)^i\tilde{A}^i\right)}_{\text{approximated personalized page rank}} \underbrace{\mathbf{F}(\Theta, X)}_{\text{feature encoder}}. \tag{3.10}$$

It first applies a feature encoder $\mathbf{F}$ on node features X and approximated personalized pagerank matrix linearly. Thereby, we have $h_i = \pi_i^{\text{ppr}}$, where π_i^{ppr} is the personalized pagerank vector. For this, we mitigate distributional shifts from graph inductive bias via instance weighting. Moreover, let $Z = \mathbf{F}(\Theta, X)$ and we can further reduce the distributional shifts from non-linear networks by the proposed discrepancy regularizer d. In our experiments, we show the application of SR-GNN on two other representative GNN models: GCN [7] and

DGI [91].

3.4 EXPERIMENTS

In this section we first describe how we create training set with a controllable amount of bias, then discuss our experiment design, demonstrate the efficacy our proposed framework for handling bias as well as its advantages over domain adaptation baselines, and finally, present a study on sensitivity to the hyperparameters.

3.4.1 Biased Training Set Creation

In order to study distribution shift in GNNs, we require a repeatable process which can generate graph-biased training sets. The core aspect of creating a biased sample for graph learning tasks requires an efficient method for finding 'nearby' nodes in the graph for a particular seed node. In this work, we use the Personalized PageRank (PPR) vectors to find such nearby nodes, $\Pi^{\text{ppr}} = (I - (1-\alpha)\tilde{A})^{-1}$. PPR vectors are well suited for this case for a number of reasons. First, several previous studies [141, 143] have shown strong correlations between the information diffusion in GNNs and PPR vectors. Second, a PPR vector can be computed for an individual node in time sublinear to the size of the graph [145] – so biased training samples can be easily generated, even for large datasets. This local algorithm provides a sparse approximation $\Pi^{\text{ppr}}(\epsilon)$ with guaranteed truncation error ϵ, such that we can efficiently compute the top-γ entries of a ppr vector with controllable residuals. Therefore, using PPR we can generate stable localized training data that can effectively challenge GNN models.

We obtain a biased sample from our scalable personalized pagerank sampler (PPR-S) as follows. For a certain label ratio τ, we compute the number of training nodes needed per label in advance. Then we repeatedly randomly select nodes that have enough neighbors in their sparse personalized pagerank vector $\pi_i^{\text{ppr}}(\epsilon)$. We add both the seed nodes and their neighbors with the same label into the training data until we have enough number of nodes for each label.

3.4.2 Experimental settings

Datasets. In our experiments, we perform semi-supervised node classification tasks on five popular benchmark datasets: `Cora`, `Citeseer`, `Pubmed` [146], `ogb-arxiv` [147] and

Reddit [10]. We use the same validation and test splits as in the original GCN paper [7] and OGB benchmark. We use the remaining nodes for training. For the unbiased baseline's performance numbers, we use a random sample from this training data. Similarly, for a biased training sample, we apply our biased sampler PPR-S on the training nodes to obtain a biased training sample and report its performance.

Baselines. Following the two scenarios outlined in Section 5.2, we consider the following methods to investigate their performance under distributional shifts: (1) Traditional GNN Models: GCN [7], GAT [11], (2) Linearized GNNs: SGC [140] and APPNP [141]. We also include methods based on unsupervised node representation learning (DeepWalk [26] and DGI [91]) as a third category (3). For these methods, we use a linear classifier learned on top of pretrained node embeddings.

Scalable biased sampler. Our scalable biased sampler uses Personalized PageRank to efficiently create biased training samples in large graphs.

Our Method. If not otherwise specified, we consider the APPNP [141] instance of Shift-Robust as our base model, and also provide two ablations of it. These ablations independently use the shift-robust techniques introduced in Section 3.3.1 and 3.3.2 to validate the effectiveness of SR-GNN.

Hyperparameters. The main hyper parameters in our sampler PPR-S are $\alpha = 0.1$,$\gamma = 100$. When the graph is large, we set $\epsilon = 0.001$ in the local algorithm for sparse PPR approximation. In SR-GNN, $\lambda = 1.0$ is the penalty parameter for the discrepancy regularizer d, the lower bound for the instance weight B_l is 0.2. For all of the GNN methods except DGI, we set the hidden dimension as 32 for Cora, Citeseer, Pubmed and 256 for ogb-arxiv, with a dropout of 0.5. In order to learn effective representations, DGI [91] usually needs a higher dimensional hidden space and so, following the DGI paper we set it as 512 across all of our experiments. We use Adam [148] as an optimizer, and set the learning rate to 0.01 and L_2 regularization to 5e-4.

Scalability. In this paper, we introduce two shift-robust techniques for GNN training: discrepancy regularization and instance reweighting. Let $\mathcal{O}(\Phi)$ be the time some GNN Φ takes to compute a single node embedding, and M be the number of training examples. The IID sample in Eq (3.4) introduces M extra forward passes and $2M$ extra backward propagation in total. Overall, the extra cost is therefore linear to and does not increase the existing asymptotic complexity. The Gaussian kernel computation in Eq (3.8) (for instance reweighting) takes $\mathcal{O}(M^2 n)$ time before training, where $h_i \in \mathbb{R}^n$. The total complexity of SR-GNN is therefore $\mathcal{O}(M\Phi + M^2 n)$. Our experiments were run on a single machine with

Table 3.1: Semi-supervised classification on three different citation networks using biased training samples. Our proposed framework (SR-GNN) outperforms **all** baselines on biased training input.

Method	Cora			Citeseer			PubMed		
	Micro-F1↑	Macro-F1↑	ΔF1 ↓	Micro-F1↑	Macro-F1↑	ΔF1 ↓	Micro-F1↑	Macro-F1↑	ΔF1 ↓
GCN (IID)	80.8 ± 1.6	80.1 ± 1.3	0	70.3 ± 1.9	66.8 ± 1.3	0	79.8 ± 1.4	78.8 ± 1.4	0
Feat.+MLP	49.7 ± 2.5	48.3 ± 2.2	31.1	55.1 ± 1.3	52.7 ± 1.3	25.2	51.3 ± 2.8	41.8 ± 6.2	28.5
Emb.+MLP	57.6 ± 3.0	56.2 ± 3.0	23.2	38.5 ± 1.2	38.6 ± 1.1	31.8	60.4 ± 2.1	56.6 ± 2.0	19.4
DGI	71.7 ± 4.2	69.2 ± 3.7	9.1	62.6 ± 1.6	60.0 ± 1.6	7.6	58.0 ± 5.3	52.4 ± 8.3	21.8
GCN	67.6 ± 3.5	66.4 ± 3.0	13.2	62.7 ± 1.8	60.4 ± 1.6	7.6	60.6 ± 3.8	56.0 ± 6.0	19.2
GAT	58.4 ± 5.7	58.5 ± 5.0	22.4	58.0 ± 3.5	55.0 ± 2.7	12.3	55.2 ± 3.7	46.0 ± 6.4	14.6
SGC	70.2 ± 3.0	68.0 ± 3.8	10.6	65.4 ± 0.8	62.5 ± 0.8	4.9	61.8 ± 4.5	57.4 ± 7.2	18.0
APPNP	71.3 ± 4.1	69.2 ± 3.4	9.5	63.4 ± 1.8	61.2 ± 1.6	6.9	63.4 ± 4.2	58.7 ± 7.0	16.4
SR-GNN **w.o.** IR	72.1 ± 4.4	69.8 ± 3.7	8.7	63.9 ± 0.7	61.8 ± 0.6	6.4	69.4± 3.4	67.6 ± 4.0	10.4
SR-GNN **w.o.** Reg.	72.0 ± 3.2	69.5 ± 3.7	8.8	66.1 ± 0.9	63.4 ± 0.9	4.2	66.4 ± 4.0	64.0 ± 5.5	13.4
SR-GNN (Ours)	**73.5 ± 3.3**	**71.4± 3.5**	**7.3**	**67.1 ± 0.9**	**64.0 ± 0.9**	**3.2**	**71.3 ± 2.2**	**70.2 ± 2.4**	**8.5**

8 CPU and 1 Nvidia T4 GPU.

3.4.3 Experiment results

We first show the performance comparison of SR-GNN (ours) and other baselines on three well-known citation GNN benchmarks in Table 3.1. We report the Micro-F1, and Macro-F1 for each method. We compare each method trained on a biased sample to a GCN trained on an unbiased sample, and report its performance drop in Micro-F1 (ΔF1). We begin by noting that when the training sample is biased, every method suffers a substantial performance drop (as indicated by the column ΔF1). However, SR-GNN consistently reduces the influence of distributional shift and decreases the performance drop (ΔF1) relative to a GCN model trained on biased input by at least 40%. We note that on these three datasets, the largest decrease in performance occurs on PubMed, where all of the existing methods experience more than a 10% absolute drop in their performance due to the biased samples. However we note that in this challenging case, the improvements of SR-GNN against APPNP (base model) also grow when the shift is larger. Finally, we see from the ablation models that the combination of both the regularization and instance reweighting appears to work better than either bias correction on its own. Our results demonstrate that our shift-robust framework is effective at minimizing the effects of distributional shift in GNN training data.

On two large benchmarks in Table 3.2 we see that the performance loss from biased sample is smaller but still significant. Even in a dense network like reddit, the localized training data still affect the GNN model performance. Compared with baselines, SR-GNN can effectively mitigate the 30% of the negative effect (Δ) relative to an unbiased GCN. When more training data is provided (5%) we can further minimize this performance gap.

Table 3.2: Semi-supervised classification on ogb-arxiv and reddit varying label ratio.

	ogb-arxiv				**reddit**			
label(%)	1 %		5 %		1 %		5 %	
Method	Accuracy	Δ ↓	Accuracy	Δ ↓	Accuracy	Δ ↓	Accuracy	Δ ↓
GCN (IID)	66.0± 0.6	0	69.1± 0.6	0	93.8 ± 0.3	0	94.0 ± 0.1	0
Feat.+MLP	45.5± 0.6	21.5	43.7± 0.3	25.4	46.6±0.6	47.2	57.2±0.2	36.8
Emb.+MLP	51.1± 1.3	14.9	56.9± 0.8	13.2	89.6 ± 0.8	4.2	90.9 ± 0.3	3.1
DGI	44.8± 3.0	21.2	49.7± 3.3	19.4	83.7±1.2	10.1	85.4±0.6	8.6
GCN	59.3± 1.2	6.7	65.3 ± 0.6	3.8	89.7±1.0	4.1	90.9±0.3	3.1
GAT	58.6± 1.0	7.4	63.4 ± 1.0	5.7	80.5±5.4	13.3	82.0±3.6	12.0
SGC	59.0± 0.7	7.0	64.2 ± 1.3	4.9	88.6±1.0	5.2	90.6±0.2	3.4
APPNP	59.8± 1.1	6.2	65.1 ± 2.6	4.0	88.4±1.0	5.4	88.9±0.8	5.1
SR-GNN **w.o.** IR	60.6± 0.2	5.4	65.1±1.8	4.0	90.4± 0.6	3.4	91.2± 0.2	2.8
SR-GNN **w.o.** Reg.	61.0± 0.3	5.0	65.8±2.0	3.3	89.4± 0.8	4.4	91.9± 0.1	2.1
SR-GNN (Ours)	**61.6±0.6**	**4.4**	**66.5±0.6**	**2.6**	**91.5± 0.5**	**2.3**	**92.1± 0.3**	**1.9**

3.5 SUMMARY

In this work we were the first to demonstrate that unbiased training data is very important for performance of GNNs. We argued that biased training data is extremely common in real world scenarios and can arise due to a variety of reasons including: difficulties of labelling large amount of data, the various heuristics or inconsistent techniques that are used to choose nodes for labelling, delayed label assignment, and other constraints from real world problems. We presented a general framework (SR-GNN) that is able to reduce influence of biased training data and can be applied to various types of GNNs, including both deeper GNNs and more recent linearized (shallow) versions of these models. With a number of experiments, we demonstrated both GNNs susceptibility to biased data and the success of our method in mitigating performance drops due to this bias: our method outperforms other GNN baselines on biased data and eliminates between $(30 - 50\%)$ of the negative effects introduced by training a GCN on biased training data.

While we have considered the general shift-robust training framework, its effectiveness on sub-graph and graph-level applications are not explored. On the sub-graph level, annotations for group anomaly detection can also be influenced by biased annotations. In the biomedical domain, the scaffold split is known to be non-IID distributed. We are interested in exploring whether the proposed approach can enhance performance in this context.

CHAPTER 4: COLLECTIVE GRAPH NEURAL NETWORKS

Knowledge graph (*e.g.* Freebase, YAGO) is a multi-relational graph representing rich factual information among entities of various types. Entity alignment is the key step towards knowledge graph integration from multiple sources. It aims to identify entities across different knowledge graphs that refer to the same real world entity. However, current entity alignment systems overlook the sparsity of different knowledge graphs and can not align multi-type entities by one single model. In this chapter, we present a **C**ollective **G**raph neural network for **Mu**lti-type entity **Align**ment, called CG-MuAlign. Different from previous work, CG-MuAlign jointly aligns multiple types of entities, collectively leverages the neighborhood information and generalizes to unlabeled entity types. Specifically, we propose novel collective aggregation function tailored for this task, that (1) relieves the incompleteness of knowledge graphs via both cross-graph and self attentions, (2) scales up efficiently with mini-batch training paradigm and effective neighborhood sampling strategy. We conduct experiments on real world knowledge graphs with millions of entities and observe the superior performance beyond existing methods. In addition, the running time of our approach is much less than the current state-of-the-art deep learning methods[3].

4.1 BACKGROUND

Knowledge Graphs (KGs) contain large volumn of relation tuples in the form of ⟨subject, *relation*, object⟩, such as ⟨Aditya Raj, *write*, Don't stop Dreaming⟩ in Figure 4.1. These relation tuples have a variety of downstream applications including Question Answering [149], Search, and Recommendation [150]. With the booming of structured and semi-structured online data, numerous knowledge graphs are extracted on the same domain [151]. Different KGs, though subject to the incompleteness in varying degrees, usually contain complementary information. Entity alignment (EA) aims to identify entities across different knowledge graphs that refer to the same real world entity. This problem also known as *entity matching/resolution* [50, 51, 53, 152] that matches records in the multi-relational databases.

In a knowledge graph, there are different entity types (*e.g.*, movie, actor, characters) and relation types (*e.g.*, *direct by*, *act by*, *release date*, *etc.*). Given the nature of entity types, the alignment strategy for different entity types could be different. For example, we observe much more characters than films, that share the same name in the IMDB-Freebase dataset. One obvious solution is to develop different models for different entity types; however, the solution

[3]A reference implementation can be found at `https://github.com/GentleZhu/CG-MuAlign`

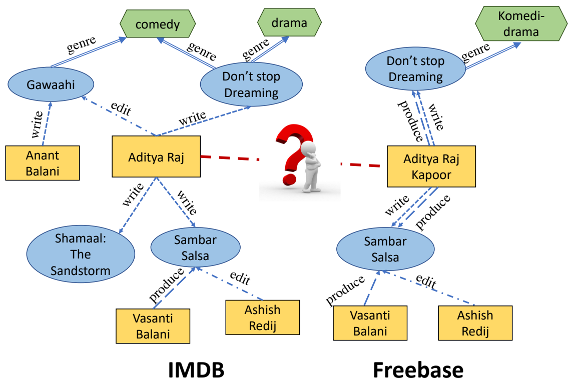


Figure 4.1: An example of Entity Alignment on person called "Aditya Raj" across IMDB and Freebase. Different edge types indicates different relations(*e.g.* "direct" and "write"). We use different color and shape indicates node types and different arrow types indicates different relations.

falls short for two reasons. First, collecting annotations and training hundreds or even more models for different entity types can be very complex and expensive. Second, an entity may belong to multiple overlapping types (*e.g.* a person can be both a movie director and a novel writer), making it hard to decide which model to apply for each entity. Thus, a multi-type entity alignment algorithm becomes critical for effective knowledge integration [153].

However, previous entity alignment methods [42, 43, 44, 45, 46, 47, 48] suffer from the following challenges presented in the multi-type entity alignment problem.

Transductive → Inductive. Previous methods [42, 43, 44, 45] adopt knowledge graph embeddings to jointly perform the KG completion and entity alignment tasks, thus may not be tuned perfectly for alignment purpose. In particular, they focus only on related entities, *i.e.* transductive setting, ignoring the potentially rich attribute information such as the name and the released date. In addition, when *new* entities are added into the graphs, these methods require complete retraining to predict alignment for new entities.

Labeled Type → Unlabeled Type. Traditional methods[50, 154] can often perform well for entity types with rich training data, but often fail for the types where training data are sparse or even lacking. Intuitively, the rich connections between different types of entities shall help boost performance for the types with small training data, but the connections are not yet effectively leveraged on a large scale.

Inspired by the recent success of Graph Neural Networks (GNN) on various tasks such as node classification [7], link prediction [16, 38] and graph classification [155], we propose to apply GNN to generate structure-aware representations for each entity, and align entities by comparing their representations. The GNN mechanism allows us to incorporate neighborhood information recursively and make inductive predictions on *unseen* entities, thus addressing both of the afore-mentioned challenges. Unfortunately, as we show in our experiments (Section. 4.4.4), a vanilla application of GNN failed terribly, obtaining only 0.33 F1 score (27% precision and 43% recall) for alignment. The key reason is that the GNN models will generate similar embeddings for the same entity from two different KGs only if both KGs contain fairly complete information about the entity. In reality, most KGs are sparse in different ways, making the embeddings often very different. For example, for the same movie, IMDB may contain editor, director and actor information, while Freebase contains only director and producer information.

This paper presents a novel GNN model that makes collective decisions [54, 156] (*i.e.* related entities alignment are determined jointly) on entity alignment for multple different types. The key of our solution is a carefully designed attention mechanism that effectively leverages shared neighborhoods as positive evidence without ignoring strong negative evidence. First, to be robust on incomplete knowledge graphs, we design the cross-graph attention that allows focusing more on the similar neighborhoods across two graphs. To illustrate the intuition, consider our motivating example in Figure 4.1. "Aditya Raj" participates in four movies in IMDB, whereas "Aditya Raj Kapoor" writes/produces two movies in Freebase; a vanilla version of GNN will generate different representations for them. Our cross-graph attention gives higher weight to shared neighbors such as "Sambar Salsa", and thus generate similar representations for the two nodes. Second, to be sensitive towards strong negative evidence, we employ relation-aware self-attention on edges that prevents blindly aligning nodes with similar neighborhoods. For example, two movies in the same series are likely to share directors, writers, and some actors; our edge-level attention allows us to pick up key differences in release year and length to distinguish them. Indeed, our experiments show that the two attention mechanisms collectively improve linkage quality by 10% F1 score in average. Although we focus on entity alignment between two graphs, entity alignment across multiple (> 2) knowledge graphs in same domain is also a very interesting and practical problem. With overlapped information, it may also provide the opportunity to potentially resolve the noise from each individual graph. We leave the exploration of the collective mechanism among multiple graphs for future work.

We note that although collectively linking entities is not a new idea [53, 54, 56, 157], our method is the first scalable solution that does not require any manually defined rules (like [56]) or logic (like [157]) for evidence propagation. Similarly, although GNN has been widely adopted for iteratively capturing the neighborhood information, our model, to the best of our knowledge, is the first that allows collective decisions in a GNN. Besides, we develop a scalable GNN framework to support large-scale entity alignment in the experiments. In Table. 4.1, we compare our method with most recent entity alignment algorithm from

Table 4.1: Comparison of methods for entity alignment. *Inductive*: Making use of node features and generalize to new nodes. *Predicate*: Modeling semantics of different relations. *Collective*: Collecting evidence from neighborhood. *Multi-type*: Handling multiple entity types in one model. *Scalable*: Scaling up to millions of nodes.

	CG-MuAlign	**MuGNN** [46]	**GCN-Align** [48]	**DeepMatcher** [50]
Inductive	✔		✔	✔
Predicate	✔	✔		✔
Collective	✔			
Multi-type	✔	✔	✔	
Scalable	✔			✔

five different perspectives. In particular, we made the following contributions.

- We propose a GNN-based knowledge graph entity alignment framework called CG-MuAlign, that collectively align entities of different types. We carefully design the attention mechanisms that can both effectively accumulate positive evidence from the neighborhood, and remain sensitive to strong negative evidence to distinguish similar but different entities.

- We scale up our model to large-scale knowledge graphs by avoiding expensive computation in each layer of the deep neural network and by relation-aware neighborhood sampling.

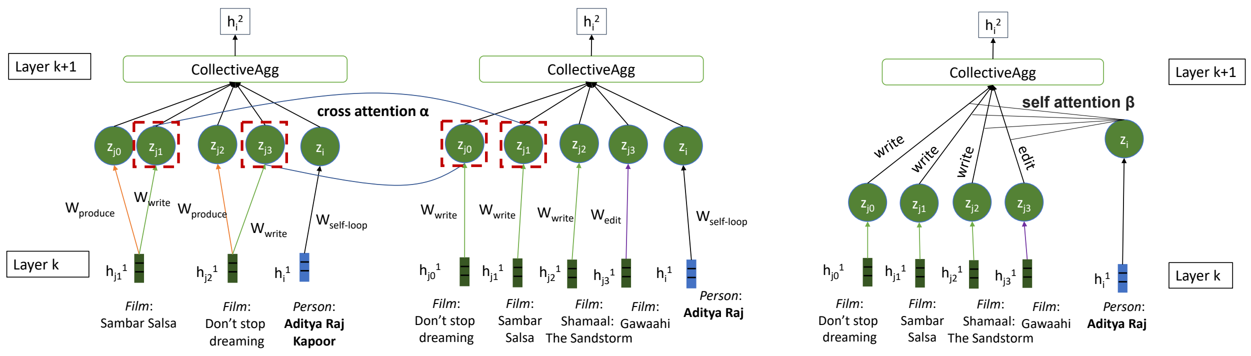
- Through extensive experiments on two different datasets, we show that our methods obtain high quality linkage (80.5% F1 and 60% recall when precision is 95%) on knowledge graphs with size of two and half millions of nodes. In particular, with the help of labeled film data, we show that CG-MuAlign trained on 2,000 person pairs can reach comparable performance with model trained on ~24,000 person pairs.

4.2 PROBLEM DEFINITION

A knowledge graph G is defined as a graph with multi-typed nodes and edges. We denote nodes $\mathcal{V}$ as entities and edges $\mathcal{E}$ as relations. Formally we have G = $(\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R})$ with a node type mapping $\phi : \mathcal{V} \rightarrow \mathcal{T}$ and edge type mapping $\psi : \mathcal{E} \rightarrow \mathcal{R}$.

Given two different knowledge graphs G and G′ on same domain, the node type and edge type are $\{\mathcal{T}, \mathcal{T}'\}$ and $\{\mathcal{R}, \mathcal{R}'\}$, respectively. Assuming node and edge types are aligned in advance: $\mathcal{T}^*\{(t, t') \in \mathcal{T} \times \mathcal{T}' | t \Leftrightarrow t'\}$, $\mathcal{R}^*\{(r, r') \in \mathcal{R} \times \mathcal{R}' | r \Leftrightarrow r'\}$, certain amount of ground truth node pairs $\mathcal{S}\{(v_i^{t^*}, v_{i'}^{t^*}) | t^* \in \mathcal{T}^*\}$ are available. Normally, there are only a few aligned seed pairs for some of the aligned node type $\mathcal{T}^*$, *i.e.* $|\mathcal{S}| \ll |\mathcal{V}|$.

Formally, we define the problem of entity alignment as follows.



(a) Cross-graph Attention (b) Relation-aware Self-Attention

Figure 4.2: Illustration of node-level and edge-level attention in CG-MuAlign

Definition 4.1 (KG Entity Alignment)**.** Given two knowledge graphs $G = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R})$ and $G' = (\mathcal{V}', \mathcal{E}', \mathcal{T}, \mathcal{R})$, *entity alignment* aims to find a set of entity pairs $\{(v_i, v_{i'}) \in \mathcal{V} \times \mathcal{V}'\}$ with high precision and recall, such that each pair refers to the same real world entity.

4.3 METHOD

CG-MuAlign features a collective GNN framework to address the KG Entity Alignment problem. Our model not only bridges the gap between single-type and multi-type alignment model, but also generalize to unlabeled types. In Section 4.3.1, we describe the overall picture of our alignment model. Then we discuss two proposed attention mechanisms and explain how they contribute to the *collective* setting in Sections 4.3.3 and 4.3.4, respectively. At last, we present our model specifications and reason about scalability concerns.

4.3.1 Solution Overview

We model the entity alignment problem as a classification problem, where we predict whether two nodes $v \in \mathcal{V}$ and $v' \in \mathcal{V}'$ represent the same real-world entity.

The model includes two GNN encoders and an entity alignment loss layer. The GNN encoder takes an K-hop sub-graph derived from target node v, aggregates the neighborhood information and outputs representation h_v^k for node v. In its k-th layer, for node i, the GNN encoder aggregates neighbor information from k-1 layer,

$$z_i^k = \textsc{Aggregate} \circ \textsc{Transform}^{(k)} \left(\{h_j^{k-1}, j \in \mathcal{N}_i\}\right) \tag{4.1}$$

where h^{k-1} is the hidden representation of the previous layers and $\mathcal{N}_i$ is the neighborhood of node i in the knowledge graph. The output representation h_i^k is the combination of h_i^{k-1} and z_i^k,

$$h_i^k = \textsc{Combine}^{(k)}\left(\{h_i^{k-1}, z_i^k\}\right) \tag{4.2}$$

For two KGs, we have two K-layer models GNN_1 and GNN_2 with identical structure and shared parameters. For each pair of entities (i, i') in the training data, we sample N negative entities from KG_1 and KG_2. Then we obtain the final representations from two GNN encoders as $(h_i^K, h_{i'}^K)$ and apply a marginal hinge loss on distance between output vector of two nodes,

$$\mathcal{L} = \sum_{(i,i')} \sum_{(i-,i'-)} \max\left(0, d(h_i^K, h_{i'}^K) - d(h_{i-}^K, h_{i-'}^K) + \gamma\right) \tag{4.3}$$

In the experiments, we use $d(x, y) = ||x - y||_2$ as the distance function.

4.3.2 Collective Graph Neural Networks

In CG-MuAlign, we first group the neighbor nodes by edge type r as $\mathcal{N}_{i,r}$ and apply different $\textsc{Transform}$, *i.e.* W_r. In Figure 4.1, for example, the target node "Aditya Raj" in the left IMDB sub-graph have $N_{i,\text{write}} = \{$Don't stop Dreaming, Shamaal: The Sandstorm, Sambar Salsa$\}$ and $N_{i,\text{edit}} = \{$Gawaahi$\}$. At each layer, we transform the neighborhood $(j \in \mathcal{N}_{i,r})$ information regarding the relation between node i and j as follows,

$$z_{i,j}^k = W_r^k h_j^{k-1}, j \in \mathcal{N}_{i,r} \tag{4.4}$$

As one entity can belong to multiple overlapping types, the above transformation explicitly differentiate the same person's representations as editor and writer in the aggregation.

We calculate node-level attention α (details in Section 4.3.3), edge-level attention β (details in Section 4.3.4) and $\textsc{Aggregate}$ neighborhood as,

$$z_i^k = \sum_{\cup N_{i,r}} \alpha_{ij}\beta_{ij} z_{i,j}^k, \Sigma_j \alpha_{ij}\beta_{ij} = 1 \tag{4.5}$$

Then we proposes the following $\textsc{Combine}$ function:

$$h_i^k = \sigma\left([W_{self}^k h_i^{k-1} || z_i^k]\right) \tag{4.6}$$

Intuitively, we concatenate the self information and neighborhood information to make the alignment decision on self information and neighborhood information independently. And we name this layer as CollectiveAgg.

In CG-MuAlign, we stack multiple layers in each GNN encoder, where the inputs at layer k is the output representation of layer k-1. The layer-0 representation is the input node features and we allow entities of different types to have different length of features. Let the hidden dimension of the model be m, we have the first layer of relation matrices $W_r^1 \in \mathbb{R}^{d_r \times \frac{m}{2}}$, where d_r is the feature length of entity in neighbor group $\mathcal{N}_r$. After concatenation as depicted in Equation 4.6, the hidden representation is then $\frac{m}{2} + \frac{m}{2} = m$. For the layer $k = 2, 3, ..., K$, we have $W_r^k \in \mathbb{R}^{m \times \frac{m}{2}}$ Then we describe how we compute the two attentions α and β.

4.3.3 Node-level Cross-graph Attention

Existing GNN-based entity alignment methods reconcile structural difference across two knowledge graphs by implicit means, such as graph matching objective [47] and rule grounding [46]. As we discussed in the introduction, the structural differences are mainly raised by the nature of incompleteness in a knowledge graph. In CG-MuAlign, we address this problem by collective aggregation of *confident* neighborhood information. Namely, we explicitly assign higher weights for those neighbors that are likely to have the corresponding ones in the other graph. We achieve this by employing a cross-graph attention mechanism that attends over the neighbor's feature vectors.

Given the candidate node pair (i, i'), we have $\mathcal{N}_i$ and $\mathcal{N}_{i'}$ as neighborhood of node i and node i', respectively. We make *soft* decisions by calculating similarity of pairs $(p, q) \in \mathcal{N}_i \times \mathcal{N}_{i'}$,

$$\alpha_p = \frac{\sum\limits_{q \in \mathcal{N}_{i'}} \exp^{-||z_p - z_q||_2}}{\sum\limits_{p \in \mathcal{N}_i} \sum\limits_{q \in \mathcal{N}_{i'}} \exp^{-||z_p - z_q||_2}}, \alpha_q = \frac{\sum\limits_{p \in \mathcal{N}_i} \exp^{-||z_q - z_p||_2}}{\sum\limits_{q \in \mathcal{N}_{i'}} \sum\limits_{p \in \mathcal{N}_i} \exp^{-||z_q - z_p||_2}} \tag{4.7}$$

The hidden representation z_p and z_q are calculated in Equation 4.4. For $p_1, p_2 \in \mathcal{N}_i$, $\alpha_{p_1} > \alpha_{p_2}$ if the accumulated similarity between p_1 and neighbors $\mathcal{N}_{i'}$ in Graph G' is larger than p_2. In computation, weight α_p and α_q are the row-wise and column-wise normalized vector for the cross-graph attention matrix $A_{i,i'} \in \mathbb{R}^{|\mathcal{N}_i| \times |\mathcal{N}_{i'}|}$. In Figure 4.2a, we turn the 1-hop neighbor in Figure 4.1 into actual computation graph in our CollectiveAgg layer. The neighborhood for "Aditya Raj" two knowledge graphs are {Gawaahi:*edit*, Don's stop Dreaming:*write* , The Sandstorm:*write*, Sambar Salsa:*write* } and {Don's stop Dreaming:*write*, Don's stop Dreaming:*produce*, Sambar Salsa:*write*, Sambar Salsa:*produce* }. The cross-graph attention will give high weights to neighbor nodes {Sambar Salsa:*write*, Don's stop Dreaming Salsa:*write*} as their hidden representation is similar. Thus, the proposed cross-graph attention leverages the *positive* evidence to the collective decisions.

Table 4.2: Alignment Example for song Radioactive. Neighbor nodes are grouped by relations as described in Section 4.3.2. Bold font indicates the neighbor node with large cross-attention weights.

	Amazon Music	Wikipedia
Attributes		
Title Duration	Radioactive 2M19S	Radioactive 2M19S
Neighbors		
Song writer	**Wayne Sermon** **A. Grant** **Dan Reynolds** **Josh Mosser**	**Wayne Sermon** **Alexander Grant** **Dan Reynolds** **Josh Mosser**
Song producer	Alex Da Kid	
Album	Night Visions (Deluxe)	Night Visions
Main performer	**Imagine Dragons** Kendrick Lamar	**Imagine Dragons**

4.3.4 Edge-level Relation-aware Self-attention

Yet, cross-graph attention neglects the *negative* evidence across the graphs. If the neighborhood aggregation only relies on the cross-attention, it fails to predict "negative" when only unimportant nodes are softly aligned. In our music data set at Figure 4.2, when aligning song "Radioactive" by American rock band Imagine Dragons between Amazon Music and Wikipedia, cross-graph attention produce *positive* evidence on most of the neighbors such as song writer, producer and one performer. However, it is an unmatched pair since the one in Amazon is a deluxe version collaborated with "Kendrick Lamar".

In other words, different relations shall play different roles in alignment prediction. For example, *performed by* is more informative than *written by.*

In fact, the computation of cross-graph attention focuses on the neighbor nodes similarity and considers each relation equally important. In light of this issue, similar with Graph Attention Networks [11], we adjust the cross-graph attention with an edge-level self-attention that considers the edge(tuple) information, *i.e.* ⟨Radioactive, *perform by*, Kendrick Lamar⟩ we calculate an edge-level self-attention by a weight vector $\vec{a}_r$ to estimate the importance of an edge composed of subject, object and relation.

$$\beta_{ij} = \frac{\exp(\sigma(\vec{a}_r^T[z_i||z_j]))}{\sum_{k\in\mathcal{N}_i}\exp(\sigma(\vec{a}_r^T[z_i||z_k]))} \tag{4.8}$$

We use $\sigma(\cdot)$ as LeakyReLU suggested in [11]. As depicted in Figure 4.2b, self-attention measures the importance of a relation tuple with the relation aware linear layer a_r. In the previous example, the attention score of ⟨Radioactive, *perform by*, Kendrick Lamar⟩ is similar with ⟨Radioactive, *perform by*, Imagine Dragons⟩ and much larger than grouped neighbors such as writer and producer.

4.3.5 Scaling up

Despite the effectiveness of the proposed GNN model, training and applying it is very expensive. We scale it up in three ways: by carefully removing unnecessary computation, by strategically sampling the neighborhood, and by selectively considering the matching candidates.

Simplifying Computation: We now analyze the effectiveness of CollectiveAgg under the Open World Assumption[4], that is, no knowledge graph has complete knowledge. We assume graph G and G' observes p portion and q portion from the underlying complete knowledge G_u. In our example in Figure 4.1, both IMDB and Freebase contains only partial information of "Aditya". Given a ground truth pair (i, i'), that both refers to the real world entity e, the number of neighborhood of e in the real world is $\mathcal{N}_e$. We now quantify the *Collective Power* by counting numer of shared (same) nodes regarding order of the neighbors.

Theorem 4.1. If G and G' have the same number of nodes, *i.e.* $|\mathcal{V}_1| = |\mathcal{V}_2|$ and there exists a injective function $\mathcal{F} : \mathcal{V}_1 \to \mathcal{V}_2$. Let K denote the order of the neighborhood, $|\mathcal{E}|$ is the total number of edges in the underlying graph G_u, the expected Collective Power decays geometrically as K increases.

$$\mathbb{E}_{(v,\mathcal{F}(v))\sim G_1}\mathrm{CP}(K) \le |\mathcal{E}| \cdot p^{\frac{K}{2}} q^{\frac{K}{2}} \tag{4.9}$$

Proof. According to the definition of p and q. Let p_i and q_i be the actual observed ratio for node v_i and $\mathcal{F}(v_i)$ in graph G and G', we have,

$$p = \frac{\sum_{i=1}^{|\mathcal{V}_1|} |\mathcal{N}_i| \cdot p_i}{|\mathcal{E}|}, q = \frac{\sum_{i=1}^{|\mathcal{V}_2|} |\mathcal{N}_i| \cdot q_i}{|\mathcal{E}|} \tag{4.10}$$

For a specific node i, the expected number of same neighborhood from a uniform distribution

[4]the assumption that the truth value of a statement may be true irrespective of whether or not it is known to be true, from wikipedia:`https://en.wikipedia.org/wiki/Open-world_assumption`

in two graphs is $|\mathcal{N}_e|p_i q_i$. Thus, when $K = 1$,

$$\mathbb{E}_{(v,\mathcal{F}(v))\sim G_1} CP(1) = \sum_i |\mathcal{N}_e| p_i q_i \tag{4.11}$$

$$\leq \sqrt{\sum_i \left(\sqrt{\mathcal{N}_e} p_i\right)^2 \sum_i \left(\sqrt{\mathcal{N}_e} q_i\right)^2} \tag{4.12}$$

$$\leq \sqrt{\sum_i \mathcal{N}_e p_i \sum_i \mathcal{N}_e q_i} = |\mathcal{E}| \cdot \sqrt{pq} \tag{4.13}$$

Recursively, we repeat the same calculation on shared neighbor nodes in previous step, that is, $\mathbb{E}[CP(K+1)] = \mathbb{E}[CP(K)] \cdot \sqrt{pq}$ QED.

The above theorem can be explained as jaccard similarity of neighborhood follows a long-tail distribution as K grows, because only same first-order neighbor nodes may contain the same second-order neighbor nodes in principle. According to this, we employ the CollectiveAgg as the Aggregate only at the last layer to reduce the computation cost as the collective power decrease. That is,

$$h_i^k = \begin{cases} \textsc{CollectiveAgg}\left(\{h_j^{k-1}, j \in \mathcal{N}_i \cup \{i\}\}\right), & k = K-1 \\ \textsc{AverageAgg}\left(\{h_j^{k-1}, j \in \mathcal{N}_i \cup \{i\}\}\right) & k < K-1 \end{cases} \tag{4.14}$$

where the AverageAgg replaces the $\alpha_{ij}\beta_{ij}$ in Equation 4.5 as $\frac{1}{|\mathcal{N}_i|}$.

Mini-batch Training and Neighborhood Sampling. Traditional graph neural nets are trained globally, which is infeasible when the graph is large. Instead, we sample a batch of positive pairs from training data and construct a K-hop sub-graph from G and G'. To further speed up the training, we adopt neighborhood sampling to control the size of the computation graph.

Lemma 4.1. Let the maximum neighborhood size as N and batch size as B, the space complexity of CG-MuAlign is $O(BN^K)$. Without batch training or sampling, the space complexity is $O(|\mathcal{V}| \cdot K)$. For training data of size S, the expected running time is $O(S \cdot N^K)$.

Additionally, we adopt a relation-aware neighborhood sampling to leverage the maximal collective power, which samples those "one-to-one" relation first. The probability of sampling possibly matched neighbor node is greater than those "one-to-many" relations. For example, one movie usually has only one director but many actors, knowing whether the director is same is more informative than knowing one actor is same. For each type of entity v^t, we calculate the average number $avg\_N_r^t$ of neighbors connected by relation r. During the neighborhood sampling process for node i of type t, we sample from the neighborhood group $\mathcal{N}_{i,r}$ with probability

$$\Pr(n) \propto \left(\frac{avg\_N_r^t}{\sum_r avg\_N_r^t} \right)^{-1} \tag{4.15}$$

Therefore, director neighbors are more likely to be sampled compared with characters and actors due to their large population. It helps make the collective decisions when we sample a small number of neighborhoods.

Candidate Generation. Though the training cost is controlled by number of GNN layers and number of sampled neighbors, the inference cost remains as a problem. Naive one-versus-all comparison leads to time complexity up to $\mathcal{O}(|\mathcal{V}|!)$. To scale up to millions of entities, we employ candidate generation during the inference stage, also known as blocking. For each test node, we use several strong keys(*e.g.* name and date of birth for person) to collect possible match entities and use CG-MuAlign to predict alignment score within candidate pairs.

4.3.6 Relations with other GNN variants

Now we summarize the key differences of proposed CollectiveAgg with previous popular GNN framework.

Similar with RGCN [13], we adopt multi-relational matrices to model the semantics of different relations when aggregating the neighbors. Our self-attention modules shares similar motivation with GAT [11]. Both GraphSage and CollectiveAgg characterize with concatenating self representation and neighborhood representations. The GraphSage GNN layer includes concatenation and aggregate function, like average

$$h_i^k = \sigma \left(W_1 \left[h_i^{k-1} || \sigma \left(W_2 \cdot \mathrm{MEAN}\{h_j^{k-1}, j \in \mathcal{N}_{i,r}\} \right) \right] \right), \tag{4.16}$$

There are two differences between CollectiveAgg and GraphSage. First, we have multi-relational projection matrix W_r in the hidden layer. Second, we use weighted average (attention) Λ instead of averaging or max pooling.

$$h_i^k = \sigma \left(W_1 \left[h_i^{k-1} || \Lambda(W_r \cdot \{h_j^{k-1}, j \in \mathcal{N}_i\}) \right] \right) \tag{4.17}$$

In the toy example below, all kinds of previous aggregation function, *e.g.* MEAN/MAX/-SUM, fail to fit the label if node id is the only input feature. A learnable mask Λ on

Table 4.3: Overall Dataset Statistics

Dataset	# Nodes	# Edges	# Node Types	# Edge Types
Movie	2,684,233	6,851,166	8	8/8
Music	1,768,983	10,723,141	6	4/5

Table 4.4: Movie Dataset

Dataset	# Films	# People	# Characters	# Genres	# Train/Test
Freebase	273,526	314,869	859,289	599	53,405/53,405
IMDB	423,118	600,909	211,895	28	

neighborhood, instead, can fit the label by masking out node c and d. To some extent, CollectiveAgg has a greater representation power for the task of entity alignment when data is sparse. For node $a \in G$ and $a' \in G'$, we have first-order neighbors $\{b, c, d\}$ in graph G and $\{b\}$ in graph G', the training label is 1.

4.4 EXPERIMENTS

We compare CG-MuAlign with other knowledge graph alignment algorithms to examine our three major claims one by one in Section 4.4.4.

- CG-MuAlign outperforms existing methods on real-world large-scale dataset.
- Collective alignment is not sensitive to the amount of training data.
- CG-MuAlign generalizes to unlabeled type effectively with limited labels.

4.4.1 Datasets

In our experiments, we use two different knowledge graph alignment data sets and evaluate the performance under inductive settings. Both (*i.e.* Movie and Music domain) contain abundant node attributes and feature with millions of nodes and tens of millions edges of different types. We report basic graph statistics in Table 6.1 and then introduce them in more details. The number of nodes and number of edges are summed over two knowledge graphs.

Movie Dataset contains a subset of IMDB (an online database of information related to films) and Freebase (a large knowledge base on general domains). The latter originally has

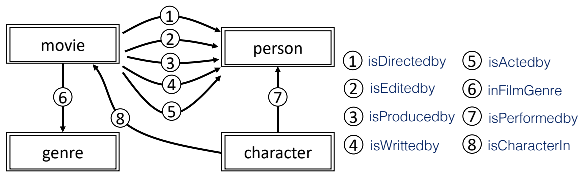


Figure 4.3: The schema of the Movie Graph

Table 4.5: Music Dataset

Dataset	# Songs	# Albums	# Artists	# Train/Test
Wikipedia	104,179	188,602	71,409	57,062/23,485
Amazon-Music	999,900	200,911	201,550	

a large number of edge types compared with IMDB. We sample a subset of Freebase that is related to the movie domain. It has ground truth links to the IMDB ID for some of the films and people. We split the ground truth pairs into training and testing data. It has four different entity types and eight different relations, the schema can be found in Figure 4.3. In Table 4.4, we report the distribution of entity types and the size of the training/testing data.

Music Dataset contains two music knowledge graph from Amazon Music and wikipedia. There are three major types in this dataset: song, album and artist. The five relations among them can be found in Figure 4.4. The positive pairs on songs and albums are generated with noise and we ask annotators to label testing pairs among a candidate pool for two types. Detailed number of entities can be found in Table 4.5.

4.4.2 Baselines

We consider methods from three families: (1) link prediction (2) entity alignment between graphs (3) entity matching in multi-relational database.

Link prediction. Between two knowledge graphs G and G', we can add *equivalent* edges

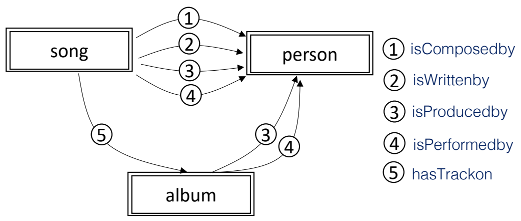


Figure 4.4: The schema of the Music Graph

between ground truth node pairs $\{(v_i^t, v_j^{t\prime})\}$. We then run advanced graph embedding algorithm with node features to embed nodes from different graphs in the same unified space. Later, we train a two-layer perceptron on the labeled *equivalent* edges. Specifically, we consider the following method that consider the node attributes:

- GraphSage [10] is the first large-scale inductive representation learning algorithm.

We denote this method as **GraphSage+NN** along with another baseline named **Feature+NN** to verify the feature effectiveness and inspect how different methods gain improvement over its performance.

Knowledge Graph Alignment. Most of the previous work focus on the transductive setting. Some recent work [46, 47, 48] based on Graph Neural Networks, start to extend graph alignment problem under inductive setting. We group these methods into **transductive only**: MuGNN [46] and BootEA [45] that both models knowledge graph embedding and entity alignment simultaneously and **inductive**: MultiKE [158] and AttrE [154] further incorporate attribute information into embedding-based entity alignment. GCN-Align [48] models both structure and attribute features with same relation matrices for different relations. As we found embedding-based methods fail to scale up to graphs with millions of entities, we carefully verify the effectiveness of proposed GNN model with following recent GNN variants.

- GCN-Align [48] models both structure and attribute features with the original graph convolutional network [7].

- GraphSage [10] concatenates the self feature vector and neighborhood aggregation vector.

- GAT [11] aggregates neighborhood information with multi-head attention.

- RGCN [13] differs GCN with multi-relational linear transformation matrices.

- R-GraphSage is a variant of GraphSage with multi-relational linear transformation matrices.

To address the scalability issue, we re-implement all of them in PyTorch [159] under DGL framework [160]. CG-MuAlign and above GNN variants adopt same mini-batch paradigm training described in Section. 4.3.5 with the batch size of 32. We sample 10 negative entities from each graph and have total 20 negative samples for every positive pair. We use Adam [148] as our optimizer with learning rate as 0.003. We set the max neighborhood size as 10 in the neighbor sampler function. The number of layers for all GNN methods are set as two. And we set hidden dimension as 100 for link prediction and graph alignment baselines.

Entity Matching. We refer methods that finds all tuple pairs (a, b) across different multi-relational databases into this category. We explore the performance of two representative methods:

- Magellan [49] is end-to-end entity matching framework that supports different matching functions like linear regression, SVM, random forest, *etc.* We choose random forest as the matching function.
- DeepMatcher [50] is a recent deep learning entity matching algorithm, we use its "hybrid" mode in our experiments.
- PARIS [156] is an unsupervised RDF ontologies alignment model, which makes collective decisions based on iterative probability estimates.

Table 4.6: Alignment Result on labeled types for inductive setting. For simplicity, transductive only methods are not included in this table. We report the standard deviation by 5-runs of each method except DeepMatcher, which takes long time for one run.

	Method	**Movie Dataset**				**Music Dataset**		
		Rec@Prec=.95	PRAUC	F1	hit@1	Rec@Prec=.95	PRAUC	F1
	Feature+NN	0	0.3672 ± 0.053	0.6380 ± 0.000	0.7197 ± 0.001	0.0025 ± 0.002	0.7251 ± 0.027	0.6795 ± 0.009
	GraphSage+NN	0.0155 ± 0.001	0.3229 ± 0.003	0.3557 ± 0.001	0.4503 ± 0.003	0.0002 ± 0.000	0.2468 ± 0.018	0.3134 ± 0.012
	Magellan	0.4387 ± 0.000	0.7067 ± 0.000	0.6945 ± 0.000	0.7974 ± 0.000	0.1071 ± 0.000	0.7461 ± 0.000	0.6760 ± 0.000
	DeepMatcher	0	0.5829 ± 0.000	0.7549 ± 0.000	0.8468 ± 0.000	0	0.1748 ± 0.000	0.3559 ± 0.000
	PARIS	0.5840 ± 0.000	0.7759 ± 0.000	0.7661 ± 0.000	0.7725 ± 0.000	0.2333	0.4175 ± 0.000	0.4640 ± 0.000
GNN variants	GCN	0.0098 ± 0.001	0.2831 ± 0.006	0.3313 ± 0.004	0.4896 ± 0.003	0.0020 ± 0.002	0.3829 ± 0.009	0.4190 ± 0.003
	GraphSage	0.1900 ± 0.007	0.5589 ± 0.004	0.5251 ± 0.003	0.6605 ± 0.009	0.2868 ± 0.029	0.8252 ± 0.003	0.7637 ± 0.001
	GAT	0.0147 ± 0.002	0.3448 ± 0.006	0.3793 ± 0.004	0.5483 ± 0.003	0.0004 ± 0.001	0.4485 ± 0.014	0.4819 ± 0.007
	RGCN	0.0106 ± 0.002	0.4247 ± 0.003	0.4435 ± 0.001	0.5450 ± 0.002	0.0025 ± 0.004	0.4419 ± 0.024	0.4625 ± 0.020
	R-GraphSage	0.2829 ± 0.009	0.6573 ± 0.003	0.6110 ± 0.004	0.7125 ± 0.003	0.4081 ± 0.029	0.8335 ± 0.004	0.7646 ± 0.003
	CG-MuAlign	**0.6010 ± 0.004**	**0.8548 ± 0.004**	**0.8050 ± 0.006**	**0.8869 ± 0.002**	**0.4437 ± 0.023**	**0.8400 ± 0.008**	**0.7762 ± 0.004**

4.4.3 Experimental Settings

Now we describe how we conduct the experiments and evaluate the performance.

Data Processing. For all of the GNN-based methods, we pre-compute the feature vector of different entities. There are two major types of features: string and numerical. We use fastText [161] to encode string features. For numerical features, we preserve the original value except time values. For time values, like duration, date of birth, we use periodical function $\sin(\cdot)$ to encode each periodical segment, *e.g.* seconds, minutes. Finally, we concatenate all of the features into a unified vector as the node feature.

For entity matching baselines, we convert one-hop neighbor node in the knowledge graph into the format **relation@attribute**, *e.g.* for a movie record, we have the field **isDirectedby@Name** indicating movie director's name. Thus, we can turn the first order information in the knowledge graph into a multi-relational table in a lossless way. In Magellan [49], the features used in the random forest are automatically generated by the model. Different string similarities are computed as features, such as jaccard similarity, levenshtein edit distance between attributes of entity pairs.

Evaluation Metrics. We evaluate different methods on both labeled and unlabeled settings and report their Recall@Precision=0.95, F1, PRAUC (precision-recall area under curve) and hit@1 on three data sets. Typically, previous research mainly use hit@1 since the evaluation data set is small. It is infeasible to conduct one-versus-all comparison when there are millions of candidate nodes. Thus, we use candidate generation introduced in Section. 4.3.5 in the testing stage and report hit@1 based on the candidate set. We report the precision and recall curve while tuning the alignment threshold. PRAUC and best F1 provide more insights how different methods perform without knowing all positive pairs. We will later show in Table 4.6, methods have similar hit@1 result could produce rather different PRAUC and F1. Besides, we propose metric Recall@Precision=0.95 to evaluate model performance when high precision is required.

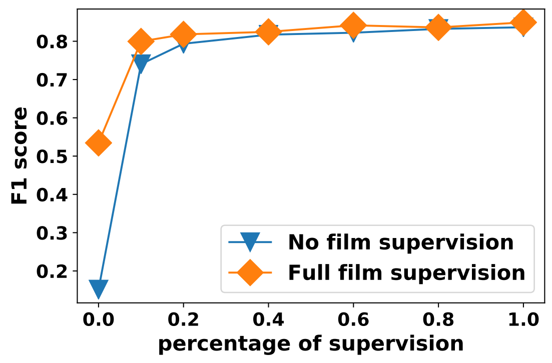
Evaluation Settings. The ground truth links between person and movie serve as positive data in training and testing. During training, we adopt the same random sampling to construct negative samples for different methods as we assume no prior knowledge of the target domain. We construct the testing data by joining output of candidate generation and the test split of ground truth links. Specifically, we use blocking function in Magellan [49] to generate candidates. For example, we use person's name and date of birth(allow missing) as the blocking key. Note that on the music domain, the ground truth links are also noisy. We annotate a subset of the candidates, thus, hit@1 metric is not included for music data. For unlabeled type evaluation, we use the same way to generate the evaluation data.

4.4.4 Experiments and Performance Study

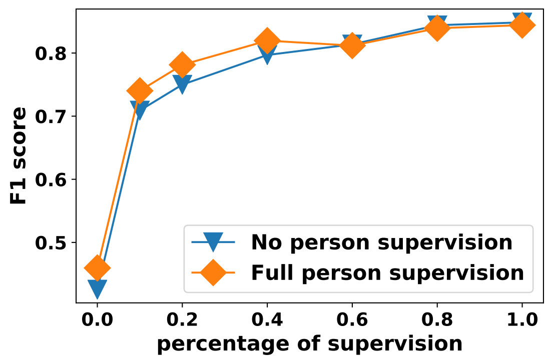
Alignment Result on labeled types. We train a unified model for multiple entity types and report all of baselines including GNN variants. From Table 4.6, we can conclude CG-MuAlign outperforms all other method. On the movie dataset, it yields a large margin over the second best method - DeepMatcher. It is mainly because IMDB and Freebase have rather different relation distributions and they suffer from data incompleteness differently. Deep-

Matcher considers the difference between attribute sets from two graphs, thus, it performs better than the remaining ones. It is quite surprising that Feature+NN outperforms most of the GNN variants, which indicates the neighborhood information affects the performance negatively in those methods. Although other GNN algorithms suffer from the sparsity of knowledge graphs while our collective aggregation layer avoid performance drop by aggregating mostly aligned neighborhood via cross-graph attention. Specifically, among three GNN variants that do not consider multi-relational structure (GCN, GrageSage, GAT) perform worse than those includes multi-relational transformation as expected. We find the concatenation mechanism first introduced in GraphSage benefit the task. The reason could be self-information is critical to the alignment task and mixing it with neighborhoods confuses the model predictions. On the music data set, CG-MuAlign gain a smaller margin over other baselines as we observe the music graphs are much denser. The performance difference is similar with the movie dataset, vanilla GNN perform badly while GraphSage and R-GraphSage obtain reasonable results compared with Feature+NN. We notice the link prediction baseline - **GraphSage+NN** achieves worse results than **Feature+NN**. GraphSage embedding models every edges of different types in the objective and the “equivalent” edges contributes than 1% in total. The “equivalent” relation may be biased by other links, therefore, predicts unsatisfactory results. Three entity matching baselines reports competitive performance on movie dataset but DeepMatcher performs much worse on the music dataset. Moreover, it achieves almost zero on the metric Rec@Prec.95. It may be caused by overfitting the noisy training data with huge amount of parameters(20 million). PARIS, though unsupervised, yield second-best F1 on the movie dataset, which proves the collective design works very nice on incomplete knowledge graphs. However, it can not tell the subtle difference between songs with slightly different names and duration due to lack of supervisions. Overall, Magellan and CG-MuAlign are the most robust methods under different datasets and metrics. As we know, the feature importance of random forest depends on the training data. So Magellan produces more false positives than ours, while CG-MuAlign reports above 50% recall when precision is at 95% across two datasets.

Sensitivity to the amount of training data. Then we investigate how much supervision needed in CG-MuAlign, since label scarcity is quite normal in the real applications. Also, we are interested in how collective alignment benefits from different types in this case. Therefore, we range ratio of supervision from 0 to 1.0 on type A and test it on type A on two conditions: (1) 100% training type of type B (2) 0% training type of type B. The result is plotted in Figure. 4.5. When we do not have any labeled data for person type, the model can achieve 0.53 F1 already and adding 10% of training data make the performance quickly converge



(a) Test on Person



(b) Test on Film

Figure 4.5: Sensitivity to the amount of training data. The orange curve indicates the collective setting, *i.e.* supervision of other types are provided. The blue curve indicates the non-collective setting.

to the final model in Figure 4.5a. Note that 10% of training data in our setting is about 2K samples only. When we have about 40% of training data, both settings are on a par with full supervision model. On the film type, the trends are similar but result is not that satisfactory when supervision is limited. We explain it as film alignment is more tricky since different films could have partially overlapped titles and same movies across different graphs could have multi-lingual names. Both figure shows that CG-MuAlign does not rely on large amount of training data.

Few-shot alignment on unlabeled types. In order to investigate the generalization capability of different methods, we design few-shot alignment experiments, that first train a model on type A and fine-tune the model with only a few (*i.e.* 2,000) training pairs of type B. The model is evaluated on the test data of new type. We train and test on person and film alternatively. Magellan and DeepMatcher are trained on one type is not compatible with the new type, so we do not report their performance in Table 4.7. In addition, we add cosine similarity of node features as an unsupervised benchmark in the table. When tested on person, we observe the cosine similarity of feature vector is a very competitive method, since people who have the same names are likely to be the same person. Another unsupervised baseline PARIS reports promising results thanks to the collective probabilistic alignment. Most of the GNN variants report poor performance except CG-MuAlign and R-GraphSage that consider the self information (person name) separately. On type of film, few-shot CG-MuAlign achieves 81.4% F1 when feature similarity only obtains 47.8%. All other methods perform worse or slightly better than node feature similarity. The result clearly demonstrates the effectiveness of collective design, especially when the training data is limited. We want to note that alignment result reported in Table 4.6 are for multi-type

Table 4.7: Alignment Result on unlabeled types for few-shot setting. We mark the best and <u>second-best</u> result. Column person stands for unlabeled type in the evaluation.

	Method	Person		Film	
		PRAUC	F1	PRAUC	F1
	Node features	0.8285	**0.8563**	0.4231	0.4780
	PARIS	0.7303	0.7489	0.8392	0.7961
GNN variants	GCN	0.3492	0.4659	0.2589	0.3223
	GraphSage	0.5495	0.6069	0.4269	0.4158
	GAT	0.3518	0.3791	0.4926	0.4818
	RGCN	0.3130	0.3518	0.4288	0.4369
	R-GraphSage	0.8065	0.7582	0.5008	0.4705
	Few-shot	<u>0.8403</u>	0.8033	<u>0.8505</u>	<u>0.8136</u>
	Fully-supervised	**0.8543**	<u>0.8214</u>	**0.9101**	**0.8794**

alignment, but here the result is evaluated for each type separately. Overall, our few-shot results are quite close to the fully-supervised version.

CHAPTER 5: TRANSFERABLE GRAPH NEURAL NETWORKS

5.1 BACKGROUND

Graph neural networks (GNNs) have been intensively studied recently [7, 162, 163, 164], due to their established performance towards various real-world tasks [10, 11, 165], as well as close connections to spectral graph theory [88, 166, 167]. While most GNN architectures are not very complicated, the training of GNNs can still be costly regarding both memory and computation resources on real-world large-scale graphs [93, 94]. Moreover, it is intriguing to transfer learned structural information across different graphs and even domains in settings like few-shot learning [168, 169, 170]. Therefore, several very recent studies have been conducted on the transferability of GNNs [1, 77, 96, 97, 98, 99, 103].

However, it is unclear in what situations the models will excel or fail especially when the pre-training and fine-tuning tasks are different. To provide rigorous analysis and guarantee on the transferability of GNNs, we focus on the setting of direct-transfering between the source and target graphs, under an analogous setting of "domain adaptation" [1, 80, 171].

In this work, we establish a theoretically grounded framework for the transfer learning of GNNs, and leverage it to design a practically transferable GNN model. Figure 5.1 gives an overview of our framework. It is based on a novel view of a graph as samples from the joint distribution of its k-hop ego-graph structures and node features, which allows us to define graph information and similarity, so as to analyze GNN transferability (§5.2). This view motivates us to design EGI, a novel GNN training objective based on ego-graph information maximization, which is effective in capturing the graph information as we define (§5.2.1). Then we further specify the requirement on transferable node features and analyze the transferability of EGI that is dependent on the local graph Laplacians of source and target graphs (§5.2.2).

All of our theoretical conclusions have been directly validated through controlled synthetic experiments (Table 5.1), where we use structural-equivalent role identification in an direct-transfering setting to analyze the impacts of different model designs, node features and source-target structure similarities on GNN transferability. In §5.3, we conduct real-world experiments on multiple publicly available network datasets. On the Airport and Gene graphs (§5.3.1), we closely follow the settings of our synthetic experiments and observe consistent but more detailed results supporting the design of EGI and the utility of our theoretical analysis. On the YAGO graphs (§5.3.2), we further evaluate EGI on the more generalized and practical setting of transfer learning with task-specific fine-tuning. We

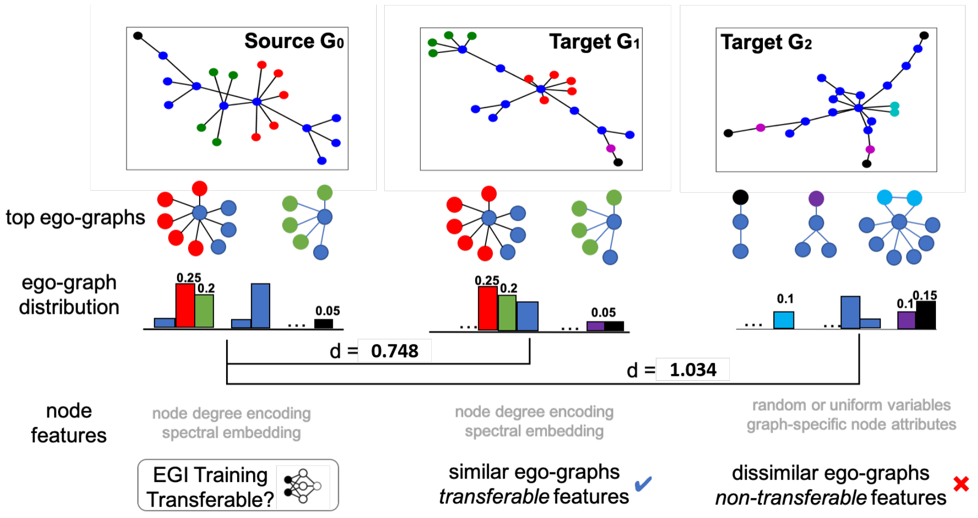


Figure 5.1: Overview of our GNN transfer learning framework: (1) we represent the toy graph as a combination of its 1-hop ego-graph and node feature distributions; (2) we design a transferable GNN regarding the capturing of such essential graph information; (3) we establish a rigorous guarantee of GNN transferability based on the node feature requirement and graph structure difference.

find our theoretical insights still indicative in such scenarios, where EGI consistently outperforms state-of-the-art GNN representation and transfer learning frameworks with significant margins.

5.2 TRANSFERABLE GRAPH NEURAL NETWORKS

In this paper, we design a more transferable training objective for GNN (EGI) based on our novel view of essential graph information (§5.2.1). We then analyze its transferability as the gap between its abilities to model the source and target graphs, based on their local graph Laplacians (§5.2.2).

Based on the connection between GNN and spectral graph theory [7], we describe the output of a GNN as a combination of its input node features X, fixed graph Laplacian L and learnable graph filters Ψ. The goal of training a GNN is then to improve its utility by learning the graph filters that are compatible with the other two components towards specific tasks.

In the graph transfer learning setting where downstream tasks are often unknown during pre-training, we argue that the general utility of a GNN should be optimized and quantified *w.r.t.* its ability of capturing the essential graph information in terms of the joint distribution of its topology structures and node features, which motivates us to design a novel ego-graph information maximization model (EGI) (§5.2.1). The general transferability of a GNN is then quantified by the gap between its abilities to model the source and target graphs. Under reasonable requirements such as using *structure-respecting* node features as the GNN input, we analyze this gap for EGI based on the structural difference between two graphs *w.r.t.* their local graph Laplacians (§5.2.2).

5.2.1 Transferable GNN via Ego-graph Information Maximization

In this work, we focus on the *direct-transfering setting* where a GNN is pre-trained on a source graph G_a in an unsupervised fashion and applied on a target graph G_b without fine-tuning.[5] Consider a graph $G = \{V, E\}$, where the set of nodes V are associated with certain features X and the set of edges E form graph structures. Intuitively, the transfer learning will be successful only if both the features and structures of G_a and G_b are similar in some ways, so that the graph filters of a GNN learned on G_a are compatible with the features and structures of G_b.

Graph kernels [172, 173, 174, 175] are well-known for their capability of measuring similarity between pair of graphs. Motivated by k-hop subgraph kernels [176], we introduce a novel view of a graph as *samples from the joint distribution of its k-hop ego-graph structures and node features.* Since GNN essentially encodes such k-hop ego graph samples, this view allows us to give concrete definitions towards *structural information* of graphs in the transfer learning setting, which facilitates the measuring of similarity (difference) among graphs. Yet, none of the existing GNN training objectives are capable of recovering such distributional signals of ego graphs. To this end, we design *Ego-Graph Information maximization* (EGI), which alternatively reconstructs the k-hop ego-graph of each center node via mutual information maximization [177].

Definition 5.1 (K-hop ego-graph)**.** We call a graph $g_i = \{V(g_i), E(g_i)\}$ a k-hop ego-graph centered at node v_i if it has a k-layer centroid expansion [176] such that the greatest distance between v_i and any other nodes in the ego-graph is k, *i.e.* $\forall v_j \in V(g_i), |d(v_i, v_j)| \leq k$, where $d(v_i, v_j)$ is the graph distance between v_i and v_j.

In this paper, we use directed k-hop ego-graph and its direction is decided by whether it is composed of incoming or outgoing edges to the center node, *i.e.*, g_i and $\tilde{g}_i$. The results apply trivially to undirected graphs with $g_i = \tilde{g}_i$.

Definition 5.2 (Structural information)**.** Let $\mathcal{G}$ be a topological space of sub-graphs, we view a graph G as samples of k-hop ego-graphs $\{g_i\}_{i=1}^{n}$ drawn *i.i.d.* from $\mathcal{G}$ with probability

[5]In the experiments, we show our model to be generalizable to the more practical settings with task-specific pre-training and fine-tuning, while the study of rigorous bound in such scenarios is left as future work.

μ, *i.e.*, $g_i \overset{i.i.d.}{\sim} \mu \; \forall i = 1, \cdots, n$. The structural information of G is then defined to be the set of k-hop ego-graphs of $\{g_i\}_{i=1}^n$ and their empirical distribution.

As shown in Figure 5.1, three graphs G_0, G_1 and G_2 are characterized by a set of 1-hop ego-graphs and their empirical distributions, which allows us to quantify the structural similarity among graphs as shown in §5.2.2 (*i.e.*, G_0 is more similar to G_1 than G_2 under such characterization). In practice, the nodes in a graph G are characterized not only by their k-hop ego-graph structures but also their associated node features. Therefore, G should be regarded as samples $\{(g_i, x_i)\}$ drawn from the joint distribution $\mathbb{P}$ on the product space of $\mathcal{G}$ and a node feature space $\mathcal{X}$.

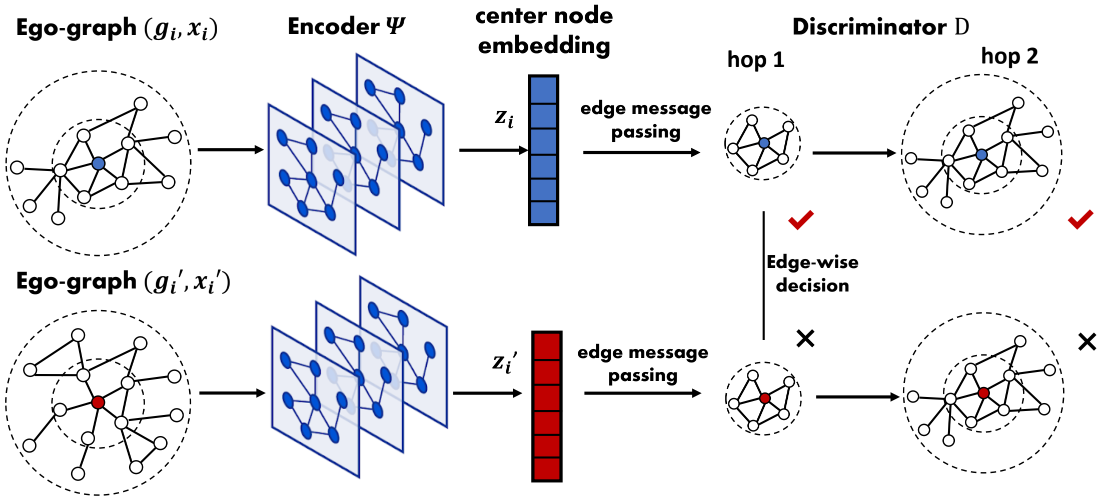


Figure 5.2: The overall EGI training framework.

Ego-Graph Information Maximization. Given a set of ego-graphs $\{(g_i, x_i)\}_i$ drawn from an empirical joint distribution $(g_i, x_i) \sim \mathbb{P}$. We aim to train an GNN encoder Ψ to maximize the mutual informaion (MI $(g_i, \Psi(g_i, x_i))$) between the defined structural information g_i[6] (*i.e.* k-hop ego-graph) and node embedding $z_i = \Psi(g_i, x_i)$. To maximize the MI, another discriminator $\mathcal{D}(g_i, z_i) : E(g_i) \times z_i \to \mathbb{R}^+$ is introduced to compute the probability of an edge e belongs to the given ego-graph g_i. We use the Jensen-Shannon MI estimator [177] in the EGI objective,

$$\mathcal{L}_{\mathsf{EGI}} = -\mathrm{MI}^{(\mathrm{JSD})}(\mathcal{G}, \Psi) = \frac{1}{N} \sum_{i=1}^{N} \left[\mathrm{sp}\left(\mathcal{D}(g_i, z_i')\right) + \mathrm{sp}\left(-\mathcal{D}(g_i, z_i)\right) \right], \tag{5.1}$$

where $\mathrm{sp}(x) = \log(1 + e^x)$ is the softplus function and (g_i, z_i') is randomly drawn from the

[6]Later in section 5.2.2, we will discuss the equivalence between $\mathrm{MI}(g_i, z_i)$ and $\mathrm{MI}((g_i, x_i), z_i)$ when node feature is structure-respecting.

product of marginal distributions, *i.e.* $z_i' = \Psi(g_{i'}, x_{i'}), (g_{i'}, x_{i'}) \sim \mathbb{P}, i' \neq i$. In general, we can also randomly draw negative g_i' in the topological space, while enumerating all possible graphs $g_{i'}$ leads to high computation cost.

In Eq. 5.1, the computation of $\mathcal{D}$ on $E(g_i)$ depends on the node orders. Following the common practice in graph generation [178], we characterize the decision process of $\mathcal{D}$ with a fixed graph ordering, *i.e.*, the BFS-ordering π over edges $E(g_i)$. $\mathcal{D} = f \circ \Phi$ is composed by another GNN encoder Φ and scoring function f over an edge sequence $E^\pi : \{e_1, e_2, ..., e_n\}$, which makes predictions on the BFS-ordered edges.

Recall our previous definition on the direction of k-hop ego-graph, the center node encoder Ψ receives pairs of (g_i, x_i) while the neighbor node encoder Φ in discriminator $\mathcal{D}$ receives $(\tilde{g}_i, x_i)$. Both encoders are parameterized as GNNs,

$$\Psi(g_i, x_i) = \text{GNN}_\Psi(A_i, X_i), \Phi(\tilde{g}_i, x_i) = \text{GNN}_\Phi(A_i', X_i), \tag{5.2}$$

where A_i, A_i' is the adjacency matrix with self-loops of g_i and $\tilde{g}_i$, respectively. The self-loops are added following the common design of GNNs, which allows the convolutional node embeddings to always incorporate the influence of the center node. $A_i = A_i'^\intercal$. The output of Ψ, *i.e.*, $z_i \in \mathbb{R}^n$, is the center node embedding, while Φ outputs representation $H \in \mathbb{R}^{|g_i| \times n}$ for neighbor nodes in the ego-graph.

Once node representation H is computed, we now describe the scoring function f. For each of the node pair $(p, q) \in E^\pi$, h_p is the source node representation from Φ, x_q is the destination node features. The scoring function is,

$$f(h_p, x_q, z_i) = \sigma\left(U^T \cdot \tau\left(W^T[h_p || x_q || z_i]\right)\right), \tag{5.3}$$

where σ and τ are Sigmoid and ReLU activation functions. Thus, the discriminator $\mathcal{D}$ is asked to distinguish a positive $((p, q), z_i)$ and negative pair $((p, q), z_i'))$ for each edge in g_i.

$$\mathcal{D}(g_i, z_i) = \sum_{(p,q) \in E^\pi} \log f(h_p, x_q, z_i), \quad \mathcal{D}(g_i, z_i') = \sum_{(p,q)}^{E^\pi} \log f(h_p, x_q, z_i'). \tag{5.4}$$

There are two types of edges (p, q) in our consideration of node orders, *type-a* - the edges across different hops (from the center node), and *type-b* - the edges within the same hop (from the center node). The aforementioned BFS-based node ordering guarantees that Eq. 5.4 is sensitive to the ordering of type-a edges, and invariant to the ordering of type-b edges, which is consistent with the requirement of our theoretical analysis on $\Delta_\mathcal{D}$. Due to the fact that the output of a k-layer GNN only depends on a k-hop ego-graph for both encoders Ψ and Φ, EGI can be trained in parallel by sampling batches of g_i's. Besides, the training objective of EGI is transferable as long as (g_i, x_i) across source graph G_a and G_b satisfies the conditions given in §5.2.2.

Connection with existing work. To provide more insights into the EGI objective, we also present it as a dual problem of ego-graph reconstruction. Recall our definition of ego-graph mutual information $\mathrm{MI}(g_i, \Psi(g_i, x_i))$. It can be related to an ego-graph reconstruction loss $R(g_i|\Psi(g_i, x_i))$ as

$$\max \mathrm{MI}(g_i, \Psi(g_i, x_i)) = H(g_i) - H(g_i|\Psi(g_i, x_i)) \leq H(g_i) - R(g_i|\Psi(g_i, x_i)). \tag{5.5}$$

When EGI is maximizing the mutual information, it simultaneously minimizes the upper error bound of reconstructing an ego-graph g_i. In this view, the key difference between EGI and VGAE [89] is they assume each edge in a graph to be observed independently during the reconstruction. While in EGI, edges in an ego-graph are observed jointly during the GNN decoding. Moreover, existing mutual information based GNNs such as DGI [91] and GMI [179] explicitly measure the mutual information between node features x and GNN output Ψ. In this way, they tend to capture node features instead of graph structures, which we deem more essential in graph transfer learning as discussed in §5.2.2.

Use cases of EGI framework. In this paper, we focus on the classical domain adaption (direct-transferring) setting [171], where no target domain labels are available and transferability is measured by the performance discrepancy without fine-tuning. In this setting, the transferability of EGI is theoretically guaranteed by Theorem 5.1. In §5.3.1, we validated this with the airport datasets. Beyond direct-transferring, EGI is also useful in the more generalized and practical setting of transfer learning with fine-tuning, which we introduced in §5.3.2 and validated with the YAGO datasets. In this setting, the transferability of EGI is not rigorously studied yet, but is empirically shown promising.

Supportive observations. In the first three columns of our synthetic experimental results (Table 5.1), in both cases of transfering GNNs between similar graphs (F-F) and dissimilar graphs (B-F), EGI significantly outperforms all competitors when using node degree one-hot encoding as transferable node features. In particular, the performance gains over the untrained GIN show the effectiveness of training and transfering, and our gains are always larger than the two state-of-the-art unsupervised GNNs. Such results clearly indicate advantageous structure preserving capability and transferability of EGI.

5.2.2 Transferability analysis based on local graph Laplacians

We now study the transferability of a GNN (in particular, with the training objective of $\mathcal{L}_{\mathsf{EGI}}$) between the source graph G_a and target graph G_b based on their graph similarity.

We firstly establish the requirement towards node features, under which we then focus on analyzing the transferability of EGI *w.r.t.* the structural information of G_a and G_b.

Recall our view of the GNN output as a combination of its input node features, fixed graph Laplacian and learnable graph filters. The utility of a GNN is determined by the compatibility among the three. In order to fulfill such compatibility, we require the node features to be *structure-respecting*:

Definition 5.3 (Structure-respecting node features)**.** Let g_i be an ordered ego-graph centered on node v_i with a set of node features $\{x^i_{p,q}\}_{p=0,q=1}^{k,|V_p(g_i)|}$, where $V_p(g_i)$ is the set of nodes in p-th hop of g_i. Then we say the node features on g_i are structure-respecting if $x^i_{p,q} = [f(g_i)]_{p,q} \in \mathbb{R}^d$ for any node $v_q \in V_p(g_i)$, where $f : \mathcal{G} \to \mathbb{R}^{d\times|V(g_i)|}$ is a function. In the strict case, f should be injective.

In its essence, Def 5.3 requires the node features to be a function of the graph structures, which is sensitive to changes in the graph structures, and in an ideal case, injective to the graph structures (*i.e.*, mapping different graphs to different features). In this way, when the learned graph filters of a transfered GNN is compatible to the structure of G, they are also compatible to the node features of G. As we will explain in Remark 5.2 of Theorem 5.1, this requirement is also essential for the analysis of EGI transferability which eventually only depends on the structural difference between two graphs.

In practice, commonly used node features like node degrees, PageRank scores [180], spectral embeddings [181], and many pre-computed unsupervised network embeddings [26, 27, 28] are all structure-respecting in nature. However, other commonly used node features like random vectors [164] or uniform vectors [182] are not and thus non-transferable. When raw node attributes are available, they are transferable as long as the concept of *homophily* [183] applies, which also implies Def 5.3, but we do not have a rigorous analysis on it yet.

Supportive observations. In the fifth and sixth columns in Table 5.1, where we use same fixed vectors as non-transferable node features to contrast with the first three columns, there is almost no transferability (see $\delta(acc.)$) for all compared methods when non-transferable features are used, as the performance of trained GNNs are similar to or worse than their untrained baselines.

With our view of graphs and requirement on node features both established, now we derive the following theorem by characterizing the performance difference of EGI on two graphs based on Eq. 5.1.

Theorem 5.1 (GNN transferability). Let $G_a = \{(g_i, x_i)\}_{i=1}^n$ and $G_b = \{(g_{i'}, x_{i'})\}_{i'=1}^m$ be two graphs, and assume node features are structure-relevant. Consider GCN Ψ_θ with k layers and a 1-hop polynomial filter ϕ.

With reasonable assumptions on the local spectrum of G_a and G_b, the empirical performance difference of Ψ_θ evaluated on $\mathcal{L}_{\mathsf{EGI}}$ satisfies

$$|\mathcal{L}_{\mathsf{EGI}}(G_a) - \mathcal{L}_{\mathsf{EGI}}(G_b)| \leq \mathcal{O}\left(\Delta_{\mathcal{D}}(G_a, G_b) + C\right). \tag{5.6}$$

On the RHS, C is only dependent on the graph encoders and node features, while $\Delta_{\mathcal{D}}(G_a, G_b)$ measures the structural difference between the source and target graphs as follows,

$$\Delta_{\mathcal{D}}(G_a, G_b) = \tilde{C}\frac{1}{nm}\sum_{i=1}^{n}\sum_{i'=1}^{m}\lambda_{\max}(\tilde{L}_{g_i} - \tilde{L}_{g_{i'}}) \tag{5.7}$$

where $\lambda_{\max}(A) := \lambda_{\max}(A^T A)^{1/2}$, and $\tilde{L}_{g_i}$ denotes the normalised graph Laplacian of $\tilde{g}_i$ by its in-degree. $\tilde{C}$ is a constant dependant on $\lambda_{\max}(\tilde{L}_{g_i})$ and $\mathcal{D}$.

Proof. The full proof is detailed in Appendix of the paper. QED.

The analysis in Theorem 5.1 naturally instantiates our insight about the correspondence between structural similarity and GNN transferability. It allows us to tell how well an EGI trained on G_a can work on G_b by only checking the local graph Laplacians of G_a and G_b without actually training any model. In particular, we define the *EGI gap* as $\Delta_{\mathcal{D}}$ in Eq. 5.7, as other term C is the same for different methods using same GNN encoder. It can be computed to bound the transferability of EGI regarding its loss difference on the source and target graphs.

Remark 5.1. Our view of a graph G as samples of k-hop ego-graphs is important, as it allows us to obtain node-wise characterization of GNN similarly as in [184]. It also allows us to set the depth of ego-graphs in the analysis to be the same as the number of GNN layers (k), since the GNN embedding of each node mostly depends on its k-hop ego-graph instead of the whole graph.

Remark 5.2. For Eq. 5.1, Def 5.3 ensures the sampling of GNN embedding at a node always corresponds to sampling an ego-graph from $\mathcal{G}$, which reduces to uniformly sampling from $G = \{g_i\}_{i=1}^n$ under the setting of Theorem 5.1. Therefore, the requirement of Def 5.3 in the context of Theorem 5.1 guarantees the analysis to be only depending on the structural information of the graph.

Supportive observations. In Table 5.1, in the $\bar{d}$ columns, we compute the average structural difference between two Forest-fire graphs ($\Delta_{\mathcal{D}}$(F,F)) and between Barabasi and Forest-fire graphs ($\Delta_{\mathcal{D}}$(B,F)), based on the RHS of Eq. 5.6. The results validate the topological difference between graphs generated by different random-graph models, while also verifying our view of graph as k-hop ego-graph samples and the way we propose based on it to characterize structural information of graphs. We further highlight in the δ(acc) columns the accuracy difference between the GNNs transfered from Forest-fire graphs and Barabasi graphs to Forest-fire graphs. Since Forest-fire graphs are more similar to Forest-fire graphs than Barabasi graphs (as verified in the $\Delta_{\mathcal{D}}$ columns), we expect δ(acc.) to be positive and large, indicating more positive transfer between the more similar graphs. Indeed, the behaviors of EGI align well with the expectation, which indicates its well-understood transferability and the utility of our theoretical analysis.

Use cases of Theorem 5.1. Our Theorem 5.1 naturally allows for two practical use cases among many others: *point-wise pre-judge* and *pair-wise pre-selection* for EGI pre-training. Suppose we have a target graph G_b which does not have sufficient training labels. In the first setting, we have a single source graph G_a which might be useful for pre-training a GNN to be used on G_b. The EGI gap $\Delta_{\mathcal{D}}(G_a, G_b)$ in Eq. 5.7 can then be computed between G_a and G_b to pre-judge whether such transfer learning would be successful before any actual GNN training (*i.e.*, yes if $\Delta_{\mathcal{D}}(G_a, G_b)$ is empirically much smaller than 1.0; no otherwise). In the second setting, we have two or more source graphs $\{G_a^1, G_a^2, \ldots\}$ which might be useful for pre-training the GNN. The EGI gap can then be computed between every pair of G_a^i and G_b to pre-select the best source graph (*i.e.*, select the one with the least EGI gap).

In practice, the computation of eigenvalues on the small ego-graphs can be rather efficient [185], and we do not need to enumerate all pairs of ego-graphs on two compared graphs especially if the graphs are really large (*e.g.*, with more than a thousand nodes). Instead, we can randomly sample pairs of ego-graphs from the two graphs, update the average difference on-the-fly, and stop when it converges. Suppose we need to sample M pairs of k-hop ego-graphs to compare two large graphs, and the average size of ego-graphs are L, then the overall complexity of computing Eq. 5.6 is $\mathcal{O}(ML^2)$, where M is often less than 1K and L less than 50. In Appendix, we report the approximated $\Delta_{\mathcal{D}}$'s *w.r.t.* different sampling frequencies, and they are indeed pretty close to the actual value even with smaller sample frequencies, showing the feasible efficiency of computing $\Delta_{\mathcal{D}}$ through sampling.

Limitations. EGI is designed to account for the structural difference captured by GNNs (*i.e.*, k-hop ego-graphs). The effectiveness of EGI could be limited if the tasks on target graphs depend on different structural signals. For example, as Eq. 5.7 is computing the average pairwise distances between the graph Laplacians of local ego-graphs, $\Delta_{\mathcal{D}}$ is possibly less effective in explicitly capturing global graph properties such as numbers of connected components (CCs). In some specific tasks (such as counting CCs or community detection) where such properties become the key factors, $\Delta_{\mathcal{D}}$ may fail to predict the transferability of GNNs. In addition, we assume similar degree of homophily between two graphs when applying EGI.

Table 5.1: Synthetic experiments of identifying structural equivalent nodes. We randomly generate 40 graphs with the Forest-fire model (F) [186] and 40 graphs with the Barabasi model (B) [187], The GNN model is GIN [182] with random parameters (baseline with only the neighborhood aggregation function), VGAE[89], DGI [91], and EGI with GIN encoder. We train VGAE, DGI and EGI on one graph from either set (F and B), and test them on the rest of Forest-fire graphs (F). Transferable feature is node degree one-hot encoding and non-transferable feature is uniform vectors.

Method	**transferable features**			**non-transferable feature**			**structural difference**	
	F-F	B-F	δ(acc.)	F-F	B-F	δ(acc.)	$\Delta_{\mathcal{D}}$(F,F)	$\Delta_{\mathcal{D}}$(B,F)
GIN (untrained)	0.572	0.572	/	0.358	0.358	/		
VGAE (GIN)	0.498	0.432	+0.066	0.240	0.239	0.001		
DGI (GIN)	0.578	0.591	-0.013	0.394	0.213	+0.181	0.752	0.883
EGI (GIN)	**0.710**	0.616	+0.094	0.376	0.346	+0.03		

5.3 REAL DATA EXPERIMENTS

Baselines. We compare the proposed model against existing self-supervised GNNs and pre-training GNN algorithms. To exclude the impact of different GNN encoders Ψ on transferability, we always use the same encoder architecture for all compared methods (*i.e.*, GIN [182] for direct-transfering experiments, GCN [7] for transfering with fine-tuning).

The self-supervised GNN baselines are GVAE [89], DGI [91] and two latest mutual information estimation methods GMI [179] and MVC [188]. As for pre-training GNN algorithms, MaskGNN and ContextPredGNN are two node-level pre-training models proposed in [77] Besides, Structural Pre-train [99] also conducts unsupervised node-level pre-training with structural features like node degrees and clustering coefficients.

Experimental Settings. The main hyperparameter k is set 2 in EGI as a common practice. We use Adam [148] as optimizer and learning rate is 0.01. All baselines are set with the default parameters. Our experiments were run on an AWS g4dn.2xlarge machine with 1 Nvidia T4 GPU. By default, we use node degree one-hot encoding as the transferable feature across all different graphs. As stated before, other transferable features like spectral and other pre-computed node embeddings are also applicable. We focus on the setting where the downstream tasks on target graphs are unspecified but assumed to be structure-relevant, and thus pre-train the GNNs on source graphs in an unsupervised fashion.[7] In terms of evaluation, we design two realistic experimental settings: (1) Direct-transfering on the more structure-relevant task of role identification without given node features to directly evaluate the utility and transferability of EGI. (2) Few-shot learning on relation prediction with task-specific node features to evaluate the generalization ability of EGI.

5.3.1 Direct-transfering on role identification

First, we use the role identification without node features in a *direct-transfering* setting as a reliable proxy to evaluate transfer learning performance regarding different pre-training objectives. Role in a network is defined as nodes with similar structural behaviors, such as *clique members*, *hub* and *bridge* [189]. Across graphs in the same domain, we assume the definition of role to be consistent, and the task of role identification is highly structure-relevant, which can directly reflect the transferability of different methods and allows us to conduct the analysis according to Theorem 5.1. Upon convergence of pre-training each model on the source graphs, we directly apply them to the target graphs and further train a multi-layer perceptron (MLP) upon their outputs. The GNN parameters are frozen during the MLP training. We refer to this strategy as *direct-transfering* since there is no fine-tuning of the models after transfering to the target graphs.

We use two real-world network datasets with role-based node labels: (1) Airport [87] contains three networks from different regions– Brazil, USA and Europe. Each node is an airport and each link is the flight between airports. The airports are assigned with external labels based on their *level of popularity*. (2) Gene [164] contains the gene interactions regarding 50 different cancers. Each gene has a binary label indicating whether it is a *transcription factor*.

The experimental setup on the Airport dataset closely resembles that of our synthetic experiments in Table 5.1, but with real data and more detailed comparisons. We train all models (except for the untrained ones) on the Europe network, and test them on all three networks. The results are presented in Table 5.2. We notice that the node degree features themselves (with MLP) show reasonable performance in all three networks, which is not surprising since the popularity-based airport role labels are highly relevant to node degrees. The untrained GIN encoder yields a significant margin over just node features, as GNN

[7]The downstream tasks are unspecified because we aim to study the general transferability of GNNs that is not bounded to specific tasks. Nevertheless, we assume the tasks to be relevant to graph structures.

encoder incorporates structural information to node representations. While training of the DGI can further improve the performance on the source graph, EGI shows the best performance there with the structure-relevant node degree features, corroborating the claimed effectiveness of EGI in capturing the essential graph information (*i.e.* recover the k-hop ego-graph distributions) as we stress in §5.2.

When transfering the models to USA and Brazil networks, EGI further achieves the best performance compared with all baselines when structure relevant features are used (64.55 and 73.15), which reflects the most significant positive transfer. Interestingly, direct application of GVAE, DGI and MVC that do not capture the input k-hop graph jointly, leads to rather limited and even negative transferrability (through comparison against the untrained GIN encoders). The recently proposed transfer learning frameworks for GNN like MaskGNN and Structural Pre-train are able to mitigate negative transfer to some extent, but their performances are still inferior to EGI. We believe this is because their models are prone to learn the graph-specific information that is less transferable across different graphs. GMI is also known to capture the graph structure and node features, so it achieves second best result comparing with EGI.

Similarly as in Table 5.1, we also compute the structural differences among three networks *w.r.t.* the EGI gap in Eq. 5.7. The structural difference is 0.869 between the Europe and USA networks, and 0.851 between the Europe and Brazil datasets, which are pretty close. Consequently, the transferability of EGI regarding its performance gain over the untrained GIN baseline is 4.8% on the USA network and 4.4% on the Brazil network, which are also close. Such observations again align well with our conclusion in Theorem 5.1 that the transferability of EGI is closely related to the structural differences between source and target graphs.

On the Gene dataset, with more graphs available, we focus on EGI to further validate the utility of Eq. 5.6 in Theorem 5.1, regarding the connection between the EGI gap (Eq. 5.7) and the performance gap (micro-F1) of EGI on them. Due to severe label imbalance that removes the performance gaps, we only use the seven brain cancer networks that have a more consistent balance of labels. As shown in Figure 5.3, we train EGI on one graph and test it on the other graphs. The x-axis shows the EGI gap, and y-axis shows the improvement on micro-F1 compared with an untrained GIN. The negative correlation between two quantities is obvious. Specifically, when the structural difference is smaller than 1, positive transfer is observed (upper left area) as the performance of transferred EGI is better than untrained GIN, and when the structural difference becomes large (> 1), negative transfer is observed. We also notice a similar graph pattern, *i.e.* single dense cluster, between source graph and positive transferred target graph G_2.

Table 5.2: Results of role identification with direct-transfering on the Airport dataset. We report mean and standard deviation over 100 runs. The scores marked with ** passed t-test with $p < 0.01$ over the second runners.

Method	**Airport** [87]		
	Europe	USA	Brazil
features	0.528±0.052	0.557±0.028	0.671±0.089
GIN (random-init)	0.558±0.050	0.616±0.030	0.700±0.082
GVAE (GIN) [89]	0.539±0.053	0.555±0.029	0.663±0.089
DGI (GIN) [91]	0.578±0.050	0.549±0.028	0.673±0.084
Mask-GIN [77]	0.564±0.053	0.608±0.027	0.667±0.073
ContextPred-GIN [77]	0.527±0.048	0.504±0.030	0.621±0.078
Structural Pre-train [99]	0.560±0.050	0.622±0.030	0.688±0.082
MVC [188]	0.532±0.050	0.597±0.030	0.661±0.093
GMI [179]	0.581±0.054	0.593±0.031	0.731±0.107
EGI (GIN)	**0.592±0.046****	**0.646±0.029** **	**0.732±0.078**

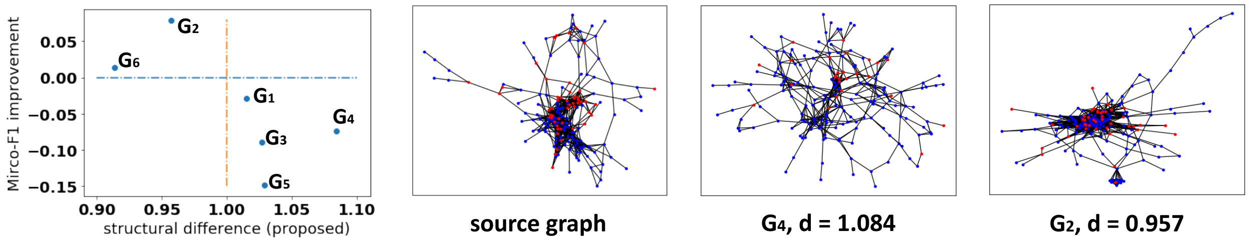


Figure 5.3: Transfer learning performance of role identification on the Gene dataset. We visualize the source graph G_0 and two example target graphs that are relatively more different (G_4) or similar (G_2) with G_0.

5.3.2 Few-shot learning on relation prediction

Here we evaluate EGI in the more generalized and practical setting of *few-shot learning* on the less structure-relevant task of relation prediction, with task-specific node features and fine-tuning. The source graph contains a cleaned full dump of 579K entities from YAGO [190], and we investigate 20-shot relation prediction on a target graph with 24 relation types, which is a sub-graph of 115K entities sampled from the same dump. In *post-fine-tuning*, the models are pre-trained with an unsupervised loss on the source graph and fine-tuned with the task-specific loss on the target graph. In *joint-fine-tuning*, the same pre-trained models are jointly optimized *w.r.t.* the unsupervised pre-training loss and task-specific fine-tuning loss on the target graph. In Table 5.3, we observe most of the existing models fail to transfer across pre-training and fine-tuning tasks, especially in the *joint-fine-tuning* setting. In particular, both Mask-GIN and ContextPred-GIN rely a lot on task-specific fine-tuning, while EGI focuses on the capturing of similar ego-graph structures that are transferable across graphs. The mutual information based method GMI also demonstrates considerable transferability and we believe the ability to capture the graph structure is the key to the transferability. As a consequence, EGI significantly outperforms all compared methods in both settings.

Table 5.3: Performance of few-shot relation prediction on YAGO. The scores marked with ** passed t-test with $p < 0.01$ over the second best results.

Method	**post-fine-tuning**		**joint-fine-tuning**	
	AUROC	MRR	AUROC	MRR
No pre-train	0.687±0.002	0.596±0.003	N.A.	N.A.
GVAE	0.701±0.003	0.601±0.007	0.679±0.004	0.568±0.008
DGI	0.689±0.011	0.586±0.025	0.688±0.012	0.537±0.023
MaskGNN	0.713±0.009	0.631±0.015	0.712±0.005	0.560±0.010
ContextPredGNN	0.692±0.030	0.662±0.030	0.705±0.011	0.575±0.021
GMI	0.728±0.005	0.625±0.009	0.721±0.007	0.643±0.011
Structural Pre-train	OOM	OOM	OOM	OOM
MVC	OOM	OOM	OOM	OOM
EGI	**0.739± 0.009****	**0.670±0.014**	**0.787 ± 0.011****	**0.729 ± 0.016****

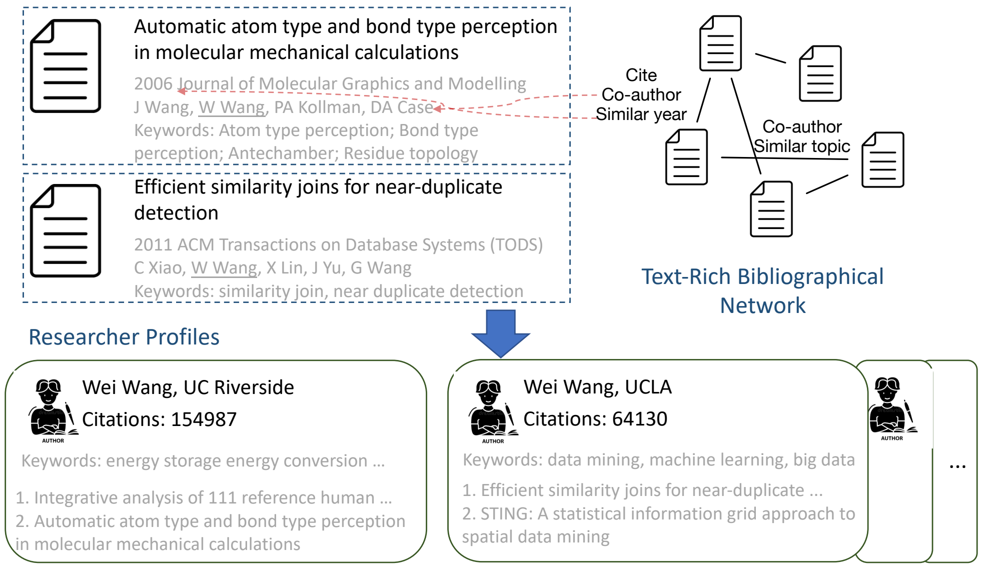
5.4 SUMMARY

To the best of our knowledge, this is the first research effort towards establishing a theoretically grounded framework to analyze GNN transferability, which we also demonstrate to be practically useful for guiding the design and conduct of transfer learning with GNNs. For future work, it is intriguing to further apply EGI to real-world applications. We believe a foundation model on molecular graphs can benefit from our approach regarding the definition of data distribution on k-hop ego-graph because lots of the target graphs share similar node features (*i.e.* atoms) and similar structural patterns (*i.e.* small Δ_D). Besides, it is also important to protect the privacy of pre-training data to avoid potential negative societal impacts.

CHAPTER 6: GRAPH-ENHANCED LANGUAGE MODEL FINE-TUNING

6.1 BACKGROUND

The rise of the academic search engines [191, 192, 193] greatly facilitate modern research activities, which let researchers efficiently retrieve relevant work and researchers in related problems. Platforms like Google Scholar, Semantic Scholar [191], PubMed [192] curate massive amount of research publications and profiles. Meanwhile, there are thousands of researchers share same or very similar first and last names. According to Google Scholar[8], there are at least 35 "James White" and 14 "Michael Jordan" create their profiles. When new research paper authored by a common name, how to accurately perform name disambiguation remains an open problem.



Automatic atom type and bond type perception in molecular mechanical calculations
2006 Journal of Molecular Graphics and Modelling
J Wang, W Wang, PA Kollman, DA Case
Keywords: Atom type perception; Bond type perception; Antechamber; Residue topology
Efficient similarity joins for near-duplicate detection
2011 ACM Transactions on Database Systems (TODS)
C Xiao, W Wang, X Lin, J Yu, G Wang
Keywords: similarity join, near duplicate detection
Cite
Co-author
Similar year
Co-author
Similar topic
Text-Rich Bibliographical Network
Researcher Profiles
Wei Wang, UC Riverside
Citations: 154987
Keywords: energy storage energy conversion ...
1. Integrative analysis of 111 reference human ...
2. Automatic atom type and bond type perception in molecular mechanical calculations
Wei Wang, UCLA
Citations: 64130
Keywords: data mining, machine learning, big data
1. Efficient similarity joins for near-duplicate ...
2. STING: A statistical information grid approach to spatial data mining
...

Figure 6.1: Illustration of Author Name Disambiguation

In our toy example (Figure 6.1), name disambiguation [68] aims to group publications of same real-world entity into separate profiles given a surface name "Wei Wang". Accurate AND is the key component of bibliographic database, which support research profiling, biobliometric calculation and *etc.*

To determine the real person of each mention in publications with the same name, the common practice is to group document into different blocks based on same name string first; Then people transform different features such as co-author and references of the document into a dense vector; Finally they perform a hierarchical linkage-based clustering considering

[8]`https://scholar.google.com/`

pairwise similarity between documents in the same block. In these steps, representation learning of documents is deemed to be crucial towards accurate AND. Given a document pair, there are two main types of information that are important to the AND-specific representation learning: (1) *textual information*: the similarity of the content and (2) *relational information*: the relevance of the meta information such as co-authors, references, venues and *etc.* Despite this problem has been studied for decades in literature [63, 65] there are two disadvantages based on our observation.

Lack of Unified Modeling. Most of existing approaches only models textual similarity or relations between documents. For example, GHOST [67] utilizes only *coauthorship* relation and AMiner-AND [63] encode the text through contrastive learning. [65] started to incorporates both factors into decisions through careful feature engineering. Yet, the possible correlation between textual and relational features between documents are not explored. For example, considering textual similarity of two paper with multiple same references may improve the accuracy.

Limited Out-of-domain Generalization. Recently, fine-tuning a pre-trained language model is the de facto method for various text related applications. However, the task of author name disambiguation requires accurate prediction on out-of-domain names. In our experiment, we found traditional contrastive fine-tuning can not improve the pairwise accuracy when the supervision is limited. Our observation confirms those from a recent study by [194].

As a response, we propose a novel framework GAND for author name disambiguation to cope with the aforementioned challenges. In our framework, meta information and texts are transformed into a text-rich bibliographical network as shown in the right side of Figure 6.1. Furthermore, there are two major differences of GAND and existing AND algorithms: (1) A *unified* representation learning module is consisted of a graph attention layer [11] and a pre-trained language model [8], which learns graph attention model for neighbors regarding text similarity derived from bottom PLM. In this way, noisy or less effective meta information would not bias the model predictions. For example, paper from different authors still likely cite popular or fundamental paper such as "BERT" and "Adam". These meta-information will hurt the accuracy of AND algorithms. Meanwhile, we still fine-tune the PLM to learn domain-specific knowledge from training data. (2) A *multi-task* training objective is devised to overcome the over-fitting issue of traditional supervised contrastive learning. Instead, we learn a classification head for each name during training and this head can be interpreted as cluster anchor embeddings. Consequently, it will not push embeddings of positive document pairs very closely as long as they are classified to the same author. In our experiments, we compare GAND with other AND algorithm and state-of-the-art fine-tuning language model methods on two name disambiguation datasets. Our framework outperforms baselines on four different clustering metrics by a clear margin. In addition, we observe consistent improvements on other graph neural network and language model baselines by employing the *multi-task* loss (see also Figure).

Our contribution could be summarized as: (1) we propose the an end-to-end framework unifying textual and relational information; (2) a multi-task objective is invented for improving the out-of-domain generalization; (3) Extensive experiments on two different AND datasets demonstrates the effectiveness of GAND.

6.2 PROBLEM DEFINITION

Given a collection of academia publications $\mathcal{D}$, we denote $\mathbf{A}$ as the set of ambiguous author names. In the following, a *reference* refers to a specific string name a, a *block* $\mathcal{D}_a \subset \mathcal{D}$ is the set of publications under *reference* a. Each paper $d_i \in \mathcal{D}_a$ is represented by a feature list x_i including title, abstract, co-author names, venue names, *etc.* We use $\mathbb{I}(d_i)$ to represent the identity of paper i. Therefore $\mathbb{I}(d_i) = \mathbb{I}(d_j)$ if two paper d_i, d_j are authored by the same identity (*i.e.* real-world person). Assuming there are k different identities in block $\mathcal{D}_a$, author name disambiguation aims to partition $\mathcal{D}_a$ into k disjoint clusters $\mathcal{D}_a^1 \cup \mathcal{D}_a^2 ... \cup \mathcal{D}_a^k = \mathcal{D}_a$.

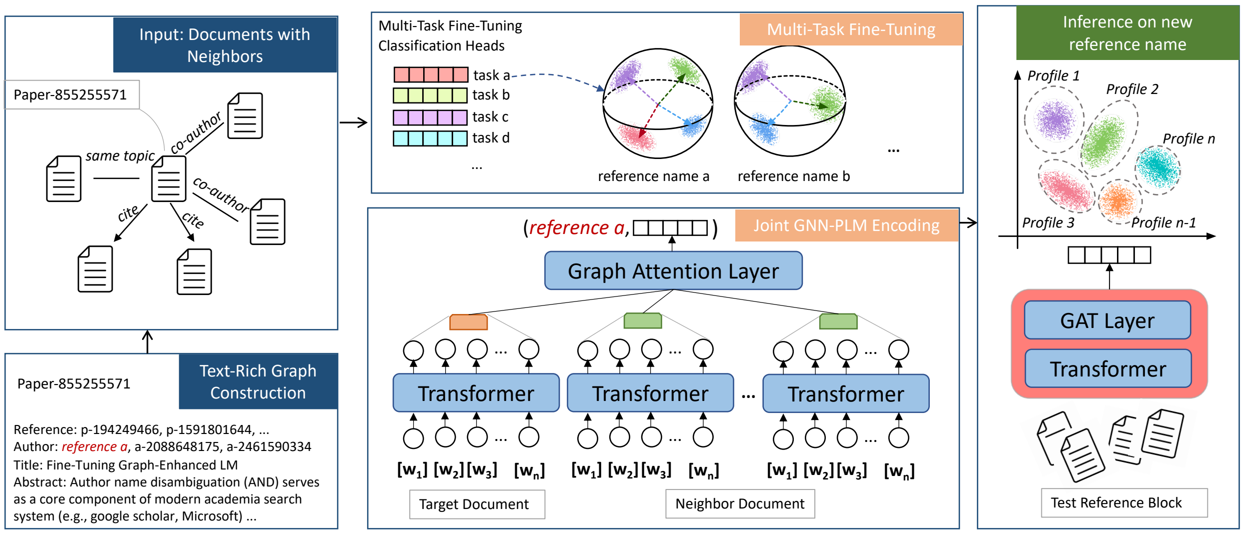
In practice, an AND algorithm needs to be able to recognize different identities for names that are absent in training data. Hereby, we formulate a multi-task learning AND problem, which shares the same setting with most of the related work [63, 65].

6.3 METHOD

6.3.1 Framework Overview

The purpose of representation learning for author name disambiguation is to embed documents from same author closely such that off-the-shelf clustering can distinguish different identities easily. However, most of the existing studies conduct pairwise or constructive learning on either textual semantics or structural attributes. [65] concatenate both attributes into a feature vector ignoring the fact that structural attributes can be noisy.

As shown in Figure 6.2, given the token sequence $\{w_1, ..., w_t\}$ of target document d_i, we adopt mean pooling of last layer of token embeddings from a pre-trained PLM as intermediate



Input: Documents with Neighbors
Paper-855255571
same topic
co-author
co-author
cite
cite
Paper-855255571
Text-Rich Graph Construction
Reference: p-194249466, p-1591801644, ...
Author: reference a, a-2088648175, a-2461590334
Title: Fine-Tuning Graph-Enhanced LM
Abstract: Author name disambiguation (AND) serves as a core component of modern academia search system (e.g., google scholar, Microsoft) ...
Multi-Task Fine-Tuning Classification Heads
Multi-Task Fine-Tuning
task a
task b
task c
task d
reference name a
reference name b
(reference a,
Joint GNN-PLM Encoding
Graph Attention Layer
Transformer
Transformer
Transformer
[w1] [w2][w3] [wn]
Target Document
Neighbor Document
Inference on new reference name
Profile 1
Profile 2
Profile n
Profile 3
Profile n-1
GAT Layer
Transformer
Test Reference Block

Figure 6.2: Framework Overview

document embedding $\mathbf{h}_i = \text{MEAN}\{h_1^{(l)}, ..., h_t^{(l)}\}$,

$$\{h_1^{(l)}, ..., h_t^{(l)}\} = \mathbf{PLM}\{w_1, ..., w_t\}, \tag{6.1}$$

Our main intuition is that only relevant neighbors can help embed target documents and textual similarity can filter out those irrelevant and noisy neighbors. In our previous example, if neighbor document $d_j \in \mathcal{N}_i$ (*e.g.* BERT) is less similar with d_i compared with other neighbors, the weight of d_j should be lower than others.

Therefore, we employ attention weights α following graph attention networks [11],

$$\alpha_{i,j} = \frac{\exp\left(\sigma\left(\mathbf{a}^{\top}[\boldsymbol{\Theta}\mathbf{h}_i \,\|\, \boldsymbol{\Theta}\mathbf{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}(i)\cup\{i\}} \exp\left(\sigma\left(\mathbf{a}^{\top}[\boldsymbol{\Theta}\mathbf{h}_i \,\|\, \boldsymbol{\Theta}\mathbf{h}_k]\right)\right)}, \tag{6.2}$$

where the activation function is LeakyReLU. The final representation of document d_i is,

$$\mathbf{z}_i = \alpha_{i,i}\boldsymbol{\Theta}\mathbf{h}_i + \sum_{j\in\mathcal{N}(i)} \alpha_{i,j}\boldsymbol{\Theta}\mathbf{h}_j. \tag{6.3}$$

6.3.2 Multi-task GNN-PLM Fine-Tuning

The proposed GAND model turn textual and structural attributes into a unified representation z_i, while how to learn a generalizable attention parameters $\{\Theta, \mathbf{a}\}$ as well as fine-tuning **PLM** remains challenging.

In **PLM** fine-tuning, the distribution shifts between training and testing corpus can lead to large performance gap [194]. In author name disambiguation, testing documents from unseen names can vary a lot from training documents from topics to domains.

To alleviate the out-of-distribution challenge, we formulate the problem as a multi-task learning problem, where each task $\mathbf{U}_M$ is consisted of documents under same surface name $\mathbf{A}_M$. Given a small amount of training task $\{\mathbf{U}_1, ..., \mathbf{U}_M\}$, each task disambiguates candidate documents $\{\mathcal{D}_1, ..., \mathcal{D}_M\}$ of a specific author name $\{\mathbf{A}_1, ..., \mathbf{A}_M\}$. For each training task, document and true author pairs $(d_i, a_i), a_i \in \mathbf{A}_M$ are provided. In each testing task $\{\mathbf{U}_{M+1}, ..., \mathbf{U}_N\}$, we will evaluate on the correctness of intra-cluster pairs (d_i, d_j) by performing a standardized agglomerative clustering.

During the training process, the model is encouraged to learn the commonalities between different names instead of memorizing the patterns of training data. Given each training sample as a triple of (document, author, task) as $\{d_i, a_i, \mathbf{U}_i\}$, we propose the following multi-task learning loss,

$$\mathcal{L} = \sum_{i=1}^{N} \left(\frac{\exp(W_{a_i}^{\mathbf{U}_i} z_i)}{\sum_{j=1}^{|\mathbf{A}_i|} \exp(W_{a_j}^{\mathbf{U}_i} z_i)} \right) \tag{6.4}$$

where $W^{\mathbf{U}_i} \in \mathbb{R}^{|\mathbf{A}_i| \times d}$ is the task-specific parameter and z_i is the normalized output of GAND. We interpreting this objective as a prototype version of traditional contrastive learning proposed for this task. Instead of contrasting with documents of other authors, the negative samples in denominator are other candidate authors.

6.4 EXPERIMENTS

6.4.1 Evaluation Setup

Datasets. We evaluate the performance of author name disambiguation on two public datasets: (1) AMiner [63] collects 600 ambiguous name references, about 203K documents are published by these authors. For each publication, there are author names, venues and keywords as meta information. (2) MAG-CS is a subset of Microsoft Academic Graph (MAG) [193] in computer science domain.[9] Each paper in MAG-CS contains its author, venue, reference information. In order to create a challenging AND task, we choose ambiguous references with at least three different real identities and each of them has at least five publications between 2000 and 2020.

[9]MAG is a large-scale collection of scientific papers in multiple academic disciplines and we select papers from top 105 venues in computer science.

Table 6.1: Overall Dataset Statistics

	AMiner	MAG-CS
# Documents	203078	6895
# Authors	6228	454
# Edges	4.99M	63.6K
# Co-author Edges	4.99M	62.2K
# Cite Edges	N/A	1463
# Total References	600	125

Compared Methods. We compare GAND with existing AND algorithms, fine-tuning language models and graph neural networks. These methods are: (1) AMiner-AND [63] learns global document embedding through contrastive learning and local graph embedding in document graph under each reference. (2) S2AND [65] construct pairwise linkage features then train a gradient boosted trees (GBT) classifier to estimate the similarity between a document pair. For pre-trained language models, we continue fine-tuning its checkpoints using a triplet margin loss as of [195]. (3) SPECTER [195] is a citation-informed language model pretraining method utilizing the citations between scientific documents. (4) OAG-BERT [196] jointly encode scientific text and venue/author information with a entity-augmented academic language model (5) SBERT [197] is state-of-the-art Sentence-BERT algorithm that unified masked and permuted pre-training for language understanding tasks. We also include two representative graph neural networks using Sentence-BERT encoded representation as node features. (6) SBERT+SGC [140] simplifies the consecutive nolinearities and weight matrices of traditional graph convolution networks [7] with best scalability. (7) SBERT+GAT [11] employs attention between target and neighbor nodes in graph and we also use this architecture in GAND. Similar with experiments in SPECTER [195], We freeze the language model embeddings in these two methods because the OOM issue of fine-tuning PLM when neighborhood size grows exponentially in GNNs. (8) GAND w.o. GNN is a variant of our approach, in which we remove all neighbors in constructed text-rich graph. (9) GAND w.o. multi-task replaces the multi-task loss in Equation 6.4 with a triplet margin loss.

Evaluation Metrics. We evaluate the clustering results of different methods on publications from test references. Suppose test document $d_i \in \mathcal{D}$, we denote the prediction $\hat{y}_i$ as a pair of reference identifier and cluster membership (r_i, c_i), and similarly the ground truth y_i as pair of reference identifier and real author id (r_i, a_i).

(1) Pairwise Micro-F1 is the harmonic mean of precision and recall between all predicted

pairs.

$$\text{Prec} = \frac{\sum_{(i,j)\in\mathcal{S}} \mathbb{I}(r_i = r_j \wedge c_i = c_j \wedge a_i = a_j)}{\sum_{(i,j)\in\mathcal{S}} \mathbb{I}(r_i = r_j \wedge c_i = c_j)}$$
$$\text{Rec} = \frac{\sum_{(i,j)\in\mathcal{S}} \mathbb{I}(r_i = r_j \wedge c_i = c_j \wedge a_i = a_j)}{\sum_{(i,j)\in\mathcal{S}} \mathbb{I}(r_i = r_j \wedge a_i = a_j)} \tag{6.5}$$

where $\mathcal{S}$ is the Cartesian product of all test documents $\mathcal{S} = \mathcal{D} \times \mathcal{D}$. Pairwise Micro-F1 measures the accuracy of all intra-cluster pairs in generated clusters (*i.e.* researcher profiles).

(2) Pairwise Macro-F1 [63] computes the average F1-score of each reference r_i according to Equation 6.5.

(3) B^3 F1 [65] is another way to compute the F1 score across all of the predictions. If we have total m clusters in test references, it computes precision and recall based intersection between ground truth clusters $\mathbf{A}$ and predicted clustering $\mathcal{C}$.

(4) Normalized Mutual Information (NMI) is a symmetric metric to measure the quality of clustering results. We average the NMI score for each reference. Specifically, for reference $r \in \mathcal{R}$, the NMI score is,

$$\text{NMI}(r) = \frac{2 \times I(Y_r; \hat{Y_r})}{[H(Y_r) + H(\hat{Y_r})]} \tag{6.6}$$

where $Y_r = \{y_i | r_i = r\}$ is the set of ground truth pairs under reference r, I is the mutual information and H is the entropy.

Experiment Settings. The input of the compared algorithms are the same documents and text-rich networks. On training references, we process the data follows refenrece implementation of each method. On test references, we will evaluate the quality of document embeddings by performing a same hierarchical clustering algorithm[10]. On AMiner dataset, we run all methods under the default training (500 references) and test (100 references) for five times. In addition, we separate 100 references from training set as validation. On MAG-CS, we perform 5-fold cross validation under five different random seeds. In each round, 20%/20% of data is used for training and validation set. Following existing studies [63], we use Macro-F1 on validation to select the best checkpoint for evaluation.

Table 6.2: Result of Author Name Disambiguation on two datasets. The scores marked with ** passed t-test with $p < 0.01$, * passed t-test with $p < 0.05$ over the second runners.

Method	MAG-CS				AMiner			
	Micro-F1	Macro-F1	B^3 F1	NMI	Micro-F1	Macro-F1	B^3 F1	NMI
AMiner-AND	0.6198	0.6514	0.6720	0.5263	0.7191	0.6726	0.7126	0.6513
SPECTER	0.7058	0.6912	0.7268	0.5355	0.6907	0.6240	0.6601	0.5558
SBERT	0.6692	0.6749	0.7088	0.5223	0.6959	0.6286	0.6643	0.5586
SBERT+SGC	0.7260	0.7399	0.7631	0.6418	0.7383	0.6772	0.7295	0.6379
SBERT+GAT	0.7325	0.7392	0.7652	0.6368	0.7499	0.6971	0.7450	0.6405
GAND **w.o.** GNN	0.7035	0.6997	0.7283	0.5780	0.7510	0.6898	0.7228	0.6414
GAND	**0.7502*****	**0.7594*******	**0.7798*******	**0.6723*******	**0.7580*******	**0.7053*******	**0.7481*****	0.6375

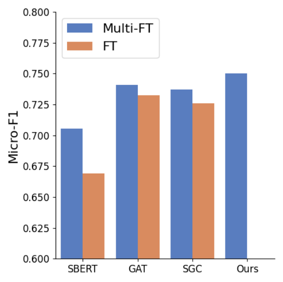
6.4.2 Results

In Table 6.2, we show the performance of all compared algorithms. We report the significance level of best result under each metric agains the second runner through two-tailed student-t test.

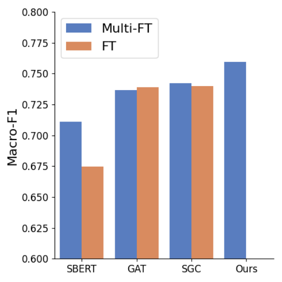
First, we observe that document representations from pre-trained language models does not outperform feature-based method (S2AND) and word embeddings (AMiner-AND) consistently. Specifically, on the smaller dataset `MAG-CS`, all three language model based approaches are better than these two methods. On the contrary, their performances are worse on `AMiner` because the overfitting issue is likely more severe when models are fine-tuned for more steps.

Effect of the multi-task training objective. In section 6.3.2, we discussed the out-of-domain challenge of fine-tuning PLMs for author name disambiguation and proposed a multi-task training objective. Besides the performance improvements we observed in Table 6.2, we apply the same objective on one PLM (*i.e.* SBERT) and GNN baselines. We have two observations from Figure 6.3. (1) Multi-Task Fine-Tuning (Multi-FT) can improve the performance of various baselines in most cases, but still worse than GAND. (2) On SBERT, Multi-FT significantly boost the performance of SBERT on all measures, which confirms our out-of-distribution observation in Section 6.3.2. Moreover, Multi-FT seems to a promising objetive for the task of author name disambiguation with PLMs.

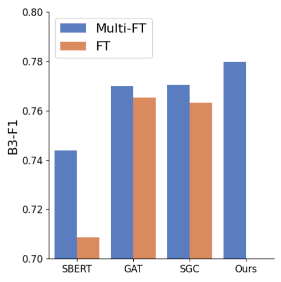
[10]`https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html`



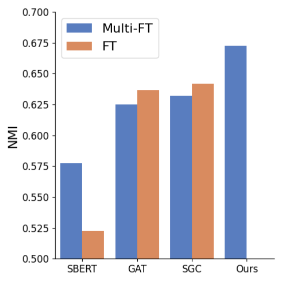
(a) Micro-F1



(b) Macro-F1

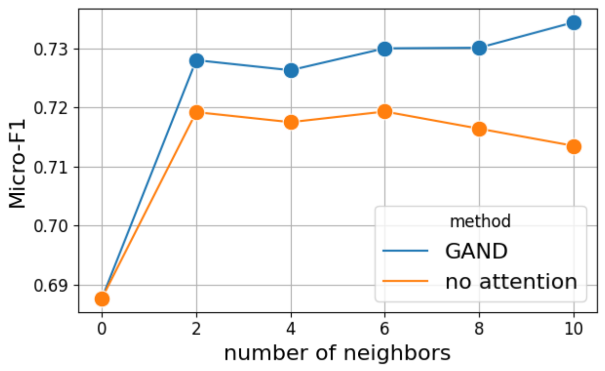


(c) B^3 F1

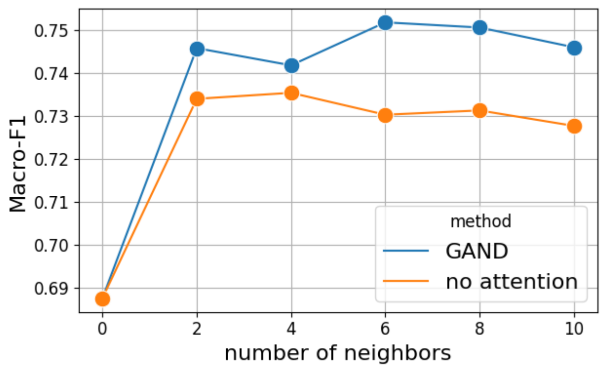


(d) NMI

Figure 6.3: Effect of Multi-Task Fine-Tuning Objective on Baselines.



(a) Effect of K on Micro-F1



(b) Effect of K on Macro-F1

Figure 6.4: Parameter study on the effect of attention and number of neighbors.

6.4.3 Parameter Study

Compared with PLMs, the additional complexity of GAND comes from the graph attention layer as shown in Equation 6.2 and Figure 6.2. So we study the effect of neighborhood size $\mathcal{N}$ and attention layer. Figure 6.4 shows the result of GAND by varying number of neighbors and replacing attention with mean pooling. We find GAND is consistently better than our variant without attention, which indicates the estimating the importance of different neighbors benefits the task. In addition, GAND with more neighbors usually perform better but not that significant. According to this result, we believe setting $K = 5$ in our main experiment is reasonable.

CHAPTER 7: GRAPH-AWARE PARAMETER EFFICIENT TUNING OF LARGE LANGUAGE MODELS

7.1 BACKGROUND

Graphs are ubiquitous in the real world [198]. In the past, graph structures have been extensively explored and utilized in many machine learning applications [199, 200]. In many practical cases, the nodes in graphs have textual features, known as Textual-Attributed Graphs (TAGs) [116]. For example, in social media [201], nodes represent users and node features are user profiles. Nodes in TAGs have both textual and structural data, which both reflect their intrinsic properties. Combining textual and structural data to modeling TAGs is an exciting new exploration for both graph machine learning and language modeling, which can benefit the application of graphs.

In TAGs, a complex correlation exists between the structural and textual data of nodes. Understanding this correlation can benefit the modeling of TAGs [202]. In Figure 7.1, user "Bob" frequently browses daily news on social media, as evidenced by the descriptions in his user profile. Users similar to Bob, who have many followers and often browse news nodes, are also likely interested in news. In other words, a graph can supplement textual attributes on a node through structural proximity. Graph Neural Networks (GNNs) are the de facto machine learning models for leveraging textual information alongside graph structures in TAGs. However, there's a lack of a unified GNN architecture compatible with different language models, especially the powerful foundation models.

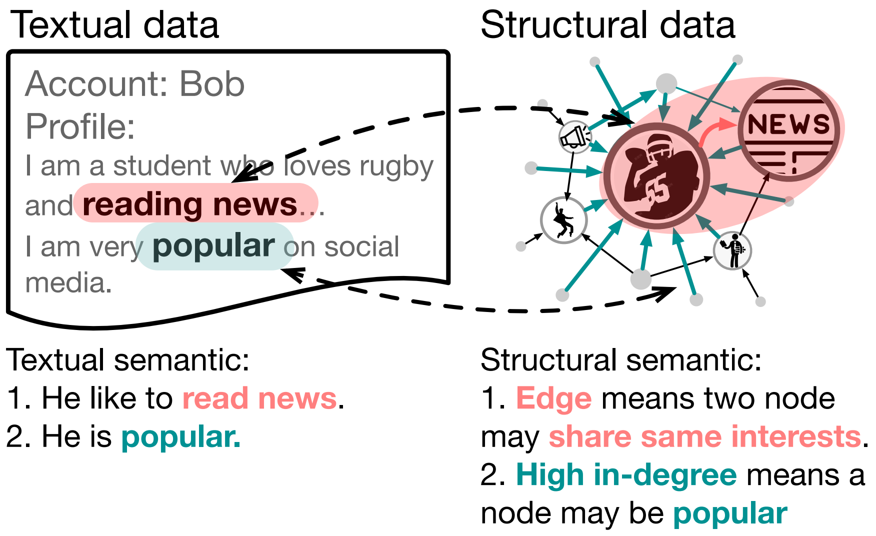

Textual data
Structural data
Account: Bob
Profile:
I am a student who loves rugby and reading news...
I am very popular on social media.
NEWS
Textual semantic:
1. He like to read news.
2. He is popular.
Structural semantic:
1. Edge means two node may share same interests.
2. High in-degree means a node may be popular

Figure 7.1: Correlation between structural and textual data of nodes in social networks.

Recently, there has been a surge in studies investigating effective ways to model both textual and structural data in TAGs. Some of these studies emphasize optimizing a cascading architecture that combines GNNs and LMs (**cascading GNN-LMs**) [116, 117]. One major challenge with these models is the extreme amount of additional computational cost brought by the message-passing mechanism. To this end, several studies have successfully reduced the memory and computational overheads of such cascaded models by freezing partial or full parameters of the backbone language models [112, 115]. Large language models exhibit superior multi-task and few-shot learning capabilities across a wide spectrum of real-world applications [110]. However, when considering cascading GNN-LMs, existing techniques cannot be scaled up to billion-scale models like Llama 2 [118] and GPT-3 [110]. Another pioneering research has ventured to fine-tune language models using unsupervised graph information (**self-supervised GNN-LMs**) [113, 114]. For instance, GIANT [113] fine-tunes language models through a neighbor prediction task, subsequently using the refined language model to extract node representations for downstream tasks. These studies have conclusively shown that graphs can indeed aid language models in comprehending textual information. However, they separate the training of GNNs and LMs, potentially leading to sub-optimal graph-aware tuning results.

Instead of using graph information as supervision, we believe graph structure can enrich textual features through language modeling. In our previous example, structural proximity can be used to infer the user's preference even if he or she does not mention it in the profile. While cascading GNNs and LLMs prove infeasible for training, we draw inspiration from works on parameter-efficient tuning of LLMs to harness the power of large language models on TAGs [128, 129, 203] Therefore, we propose the use of GNNs as adapters for LLMs (i.e., GraphAdapter) offering several advantages:

- **Lightweight:** An GNN adapter introduces fewer than 1% of the trainable parameters compared to the popular Llama 2-7B model.
- **Convenience:** Given a pre-trained LLM and unlabeled graph, one can seamlessly integrate a graph-specific adapter for multiple graph applications.
- **Graph-aware tuning:** This enhances the predictive accuracy of the fine-tuned model by leveraging graph structures.

Now we present the details of GraphAdapter with respect to pre-training and fine-tuning of the adapter GNNs. To capture the data distribution of the graph, we employ parameter-efficient tuning of LLMs on node texts. This approach is similar to the continual training of language models [204] except GNN is the tuning parameter, which helps reduce the distribution discrepancy between the pre-training corpus and target data. To further improve the efficiency, we employ the GNN adapter only at the transformer's last layer and implement residual learning for autoregressive next token prediction. Different from a traditional adapter, we perform mean-pooling on the hidden representations from a GNN adapter and LLMs, then optimize the adapter to improve the next-word prediction of the LLMs. Once the adapter is trained, one can use GraphAdapter together with the backbone LLMs on various downstream tasks. For instance, we use a classification head atop the embeddings of the last token to fine-tune for node classification.

To verify the effectiveness of GraphAdapter, we conduct extensive experiments on multiple real-world TAGs including social and citation networks.

GraphAdapter achieves an improvement of 4.7% over state-of-the-art cascaded GNN-LM methods and 5.4% over self-supervised GNN-LMs on average, with 30X fewer training parameters and storage.

Moreover, once GraphAdapter is pre-trained, it can be conveniently fine-tuned for various tasks. Our ablation analysis shows that the pre-training step consistently improves the model performance across different graphs. We summarize our contributions as follows,

- GraphAdapter is a novel approach that harnesses the large language models on graph structure data with parameter-efficient tuning.
- We propose a residual learning procedure to pre-train the GNN adapter with the LLMs. The pre-training step significantly improves the fine-tuning performance of GraphAdapter.
- We conduct extensive experiments on large-scale TAGs using state-of-the-art open-sourced large language models (GPT-2 1.5B [205] and Llama 2 13B [118]). The results demonstrate that *Graph-Adapter* can also reap the benefits of a larger model.

7.2 PROBLEM DEFINITION

Before introducing the proposed method, it's important to understand some basic concepts and the background of pre-trained language models, graph neural networks, and text-attributed graphs.

7.2.1 Pretrained Language Model

Textual data. Textual data can be formulated as $\mathbb{D} = \{d_1, d_2...d_K\}$. It can be tokenized into a sequence of tokens $\mathbb{S} = \{s_1, s_2, ..., s_L\}$, where s_i represents a specific token-id. In most cases, the first token in the sentence (i.e., s_0) is [**CLS**], indicating the beginning of this sentence.

Framework of PLMs. A PLM consists of a multi-layer transformer encoder that takes a sentence S_i as input and outputs the hidden states of each token:

$$\textbf{Transformer}(\{s_{i,0}, ..., s_{i,L}\}) = \{h_{i,0}, ..., h_{i,L}\}, \tag{7.1}$$

where $h_{i,k}$ is the dense hidden state of $s_{i,k}$.

Pre-training of PLMs. This paper uses the auto-regression task as an instance of pre-training, which is commonly applied to auto-regressive PLMs [109]. Given a sequence $\mathbb{S} = \{s_0, ..., s_L\}$, the goal is to model the joint probability of the sequence $P(\mathbb{S})$.

$$P(\mathbb{S}) = \prod_{k=1}^{L} p(s_i | s_0, ...s_{k-1}) \tag{7.2}$$

The transformer block is used to model these conditional probabilities. More specifically, at time step k ($0 < k \leq L$), the transformer receives $\{s_0...s_{k-1}\}$ and outputs their hidden states $\{h_{i,0}, ..., h_{i,k}\}$. The $h_{i,k}$ are used to predict the probability distribution of the next token.

$$p(s_i | s_0, ...s_{k-1}) = \hat{s}_k = \sigma(\textbf{Head}(h_{i,k})) \tag{7.3}$$

The model parameters are trained to maximize the likelihood of $p(\mathbb{S})$, which is equivalent to minimizing the negative log-likelihood. Therefore, the loss function is:

$$\mathcal{L}_{LM} = \sum_{k=1}^{L} \textbf{CrossEntropy}(s_k, \hat{s}_k) \tag{7.4}$$

Sentence representation. Given a sentence $\mathbb{S}$ with length L, its sentence representation W can be obtained by three methods [206, 207]: (1) first token representation, which uses the hidden state of the [**CLS**] token ($h_{i,0}$) as sentence representation. (2) mean-pooling representation, which is obtained by mean-pooling of all hidden states (i.e., $\textbf{Pool}(\{h_0...h_L\})$). (3) last token representation, which uses the hidden state of the last token.

PLMs with prompts. Due to the gap between pretraining tasks and downstream tasks, sentence representation may be hard to contain all the sentence information, thereby requiring fine-tuning for specific tasks. To address this issue, some studies utilize prompts to extract task-specific sentence features [208]. For example, suppose a $\mathbb{S}_i$ is a paper titled "Llama 2: Open Foundation and Fine-Tuned Chat Models", and the task is to classify the subject of it belongs. We can add some prompts to the sentence:

$$\{[Title], this, paper, belong, to, which, subject?\} \tag{7.5}$$

We denote this new sentence with the prompt inserted as $\mathbb{S}_{i|\mathbb{P}}$, where $\mathbb{P}$ represents the newly inserted tokens. We use the hidden state of the last token as the sentence representation, denoted as $W_{i|\mathbb{P}}$. Since the last token is used to predict the next token distribution in the pre-training stage, it can naturally combine the inserted prompt information into the original sentence and extract the prompt-related semantics. Extensive studies [203, 209] show that using prompts can reduce the gap between PLMs and downstream tasks and maximize the utilization of knowledge learned by PLMs during pre-training.

7.2.2 Graph Neural Network

Graph Neural Networks (GNNs) have achieved remarkable success in modeling graphs [11, 210]. The message-passing framework is a commonly used architecture of GNN.

Graph. Let $G = \{V, A\}$ denote a graph, where V is the node set and A is the adjacency matrix, with $A_{ij} = 1$ meaning there is an edge between node i and node j. Usually, each node i is associated with a node feature x_i^0.

Framework of GNN. The message-passing framework takes a set of node features $\mathcal{X} = \{x_i^0 | i \in V\}$, and an adjacency matrix A as input and iteratively captures neighbors' information via pooling. More specifically, for a given node $i \in V$ in the l-th layer of message-passing, it can be formulated as:

$$x_i^l = f_2(\mathbf{Pool}\{f_1(x_j^{l-1}|\theta_1^l)|j \in \mathcal{N}_i\}, x_i|\theta_2^l) \tag{7.6}$$

where $\mathbf{Pool}\{\cdot\}$ is an aggregation function that combines the features of neighboring nodes, such as mean-pooling. And $\mathcal{N}_i$ denotes the set of neighbors of node i. Besides, $f_1(\cdot|\theta_1^l)$ and $f_2(\cdot|\theta_2^l)$ denote two trainable transformations with parameters θ_1^l and θ_2^l respectively. Further, we denote an l_{max} layer message-passing framework as GNN, formally:

$$z_i = \mathbf{GNN}(x_i^0, \mathcal{X}^0, A|\Theta_g) \tag{7.7}$$

where $z_i = x_i^{l_{max}}$, and Θ_g represents all the trainable parameters in the GNN. We use z_i as the structural representation for node i.

7.2.3 Text-Attributed Graph

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{A}\}$ denote a text-attributed graph, where $\mathcal{V}$ is the node set and $\mathcal{A}$ is the adjacency matrix. Each node $i \in \mathcal{V}$ is associated with a tokenized textual data, represented by $\mathbb{S}_i = \{s_{i,0}, ..., s_{i,L_i}\}$, which represents the textual data of the node.

Problem Definition: Give a text-attributed graph $\mathcal{G}$, the problem this paper focuses on is how to efficiently utilize the unlabeled textual data $\{\mathbb{S}_i | i \in \mathcal{V}\}$ in $\mathcal{G}$ to enhance the modeling of $\mathcal{G}$ by parameter efficient tuning of PLMs. Differing from in-context learning on graphs, the motivation behind this research is to leverage LLMs for representation learning. In other words, potential users will have ownership of the model.

7.3 METHOD

This section introduces the proposed framework, referred to as GraphAdapter, which uses GNNs as adapters for LLMs to better model TAGs.

7.3.1 Overview

The core idea of GraphAdapter is: (1) combining GNNs as adapters to LMs. (2) pre-training GNN to align with LMs and enhance LMs through unlabeled textual data.

Motivation: In the textual data of TAGs, many structure-related semantics are hard to infer from context alone. As illustrated in the example in Figure 1, we can easily infer that this user is “popular” based on his degree in the social network, but it is difficult to infer from their description of habits alone. Combining structural information can enhance language models’ ability to model these structure-related semantics in TAGs. Meanwhile, the process of enhancement is learning how to model structure. Therefore, the proposed method GraphAdapter, which first uses GNN as adapters for frozen PLMs, to combine structural information with PLMs, and then pre-trains them through the semantic understanding task on TAGs.

Language-structure pre-training: In the field of natural language processing, pre-training is a common strategy used to self-supervised enhance language models’ ability for semantic understanding, with techniques such as auto-regressive pre-training (e.g., GPT-2/3 [109, 110], Llama 2 [118], etc.) and auto-encoding pre-training (e.g., BERT[211], RoBERTa[106], etc.). Following our motivation, GraphAdapter uses the same pre-training task as these PLMs. To facilitate comprehension, this section only discusses GraphAdapter based on auto-regressive pre-training, and further details on how GraphAdapter is combined with other pre-training tasks can be found in the appendix. Since the pre-training process uses the context semantic to supervise structure learning, we refer to this pre-training as language-structure pre-training.

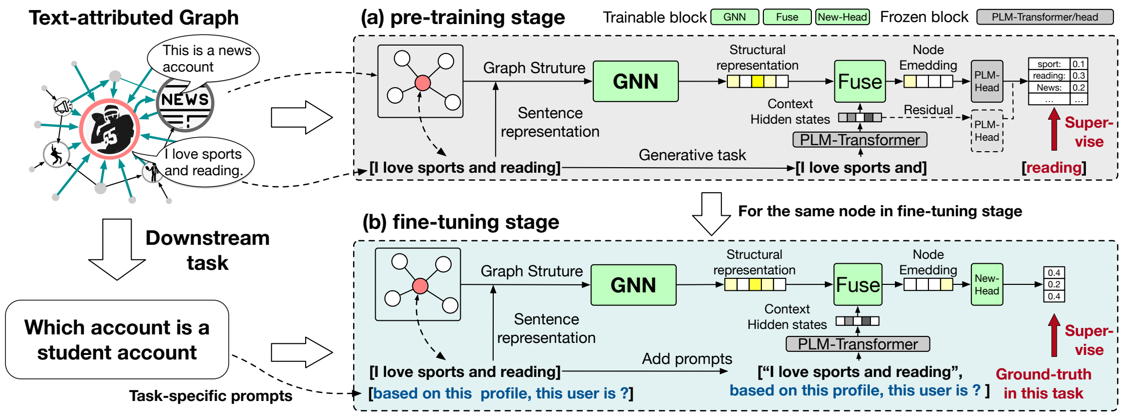


Figure 7.2: Framework of GraphAdapter. In the pre-training stage, Step 1. GNN models the node structure information, Step 2. integrates the structural information with the corresponding text fragment encoded by LM, and Step 3. predicts the masked token.

Framework: The framework of GraphAdapter is shown in Figure 7.2 (a). We also show how to fine-tune GraphAdapter on the downstream tasks in Figure 7.2 (b), we detail this part in Section 4.3. Given the textual data and graph structural data of a node, during the pre-training process, Step 1. GNN models the node structure information; Step 2. integrates the structural information with the corresponding context hidden-states modeled by PLM; and Step 3. predicts the next token. During this pre-training process, GraphAdapter can learn rich information. **Align GNN with the language model.** During the learning process, the node representation obtained by GNN is constantly combined with different representations modeled by the language model for reasoning, and the entire process naturally aligns these two. **Enhance GNN in modeling graph structure.** During the entire pre-training stage, the semantic information in the textual data supervises the GNN to model the graph structural information. **Better understanding the semantics in TAG**. GraphAapter can learn how to combine LLM and GNN to model the semantic information on TAG.

7.3.2 Pre-training on TAGs

In the training stage, GraphAdapter uses the textual data of each node in TAG to train GNN.

Pipeline of pre-training: Given a text-attributed graph $\mathcal{G}$, node i and its textual data $\mathbb{S}_i = \{s_{i,0}, ..., s_{i,L_i}\}$, GraphAdapter uses all the tokens in $\mathbb{S}_i$ as supervision. For the k-th token, GraphAdapter first extracts its previous tokens $\mathcal{S}_{i,k} = \{s_{i,0}, ..., s_{i,k-1}\}$. Then, GNN models node i's structure information z_i. The structure information is then combined with the previous tokens to predict the probability distribution of the next token, where the ground truth is token $s_{i,k}$.

Structural representation: GraphAdapter obtains its structural features z_i through GNN. Here we use a general GNN based on the message-passing framework, which continuously aggregates neighbor features to obtain the new node's structural information. For whole process is formalized as:

$$z_i = \mathbf{GNN}(x_i^0, \mathcal{X}, \mathcal{A}|\Theta_g) \tag{7.8}$$

where x_i^0 and $\mathcal{A}$ represent the initial node feature input and adjacency matrix in GNN, respectively. This paper used the sentence representation of the corresponding node as x_i^0. See more details about GNN in Section 3.2.

Context hidden-states. GraphAdapter use the pre-trained transformer in PLM to encode $\mathcal{S}_{i,k}$, it is formalized as:

$$h_{i,k} = \mathbf{Transformer}(\{s_{i,0}, s_{i,1}, ..., s_{i,k-1}\}) \tag{7.9}$$

Where the **Transformer**'s parameters are trained in frozen, and $h_{i,k}$ is the context hidden-states $\mathcal{S}_{i,k}$. Note that in the pretraining stage of PLM, $h_{i,k}$ is directly used to predict the next token, so $h_{i,k}$ contains both the context information and a certain of PLMs' prediction result.

Fusion block: GraphAdapter next fuse structural representation into context hidden-states, which is formalized as:

$$r_{i,k} = \mathbf{Fusion}(h_{i,k}, z_i|\Theta_{fuse}), \tag{7.10}$$

The **Fusion**$(*)$ function is trainable with parameters Θ_{fuse}. In this paper, MLPs are used as the structure of fusion. The process involves concatenating $h_{i,k}$ and z_i, and then feeding the resulting vector into MLPs.

Residual connection: the fused $r_{i,k}$ contains both structure information and context information. However, not every token's prediction requires the graph structure. For example, in the sentence "This paper focuses on graphs," the word "on" is simply a fixed collocation and easily inferred by context. Intuitively, words related to graph structure should be difficult for the language model to predict based on context. Therefore, the results of pre-trained language models are reused. We separately calculated the prediction probabilities of the language model alone and the probabilities that mixed the graph structure and the previous predictions. The two probabilities are then averaged to obtain the final prediction result. Formally:

$$\hat{s}_{i,k}^{LM} = \sigma(\mathbf{Head}(h_{i,k})), \hat{s}_{i,k}^{GNN} = \sigma(\mathbf{Head}(r_{i,k})) \tag{7.11}$$

$$\hat{s}_{i,k}^{ALL} = (\hat{s}_{i,k}^{LM} + \hat{s}_{i,k}^{GNN})/2 \tag{7.12}$$

Where σ denotes the softmax function. Adding the original language model prediction results allows GNN to focus more on words that the language model cannot understand well.

Optimization: Our goal is to minimize the cross-entropy loss between the predicted probability distribution and the ground-truth distribution. Formally,

$$\mathcal{L}_{i,k} = \mathbf{CrossEntropy}(\hat{s}_{i,k}^{ALL}, s_{i,k}) \tag{7.13}$$

$$\min_{\Theta_g, \Theta_{fuse}} \sum_{i \in V} \sum_{k \in \mathcal{S}_i} \mathcal{L}_{i,k} \tag{7.14}$$

Note, only $\mathbf{GNN}(*|\Theta_G)$ and $\mathbf{Fusion}(*|\Theta_{fuse})$ of GraphAdapter are trainable in whole pre-training.

GNN as Adapter: In the whole pre-training stage, the GNN combines with the frozen LM's hidden states outputted from the transformer block. The combined hidden states are then input into the PLM's prediction head. Thus, the GNN acts as an adapter, altering the language model's predictions. Since the hidden states outputted by the transformer block can be pre-processed and stored in advance. Therefore, the entire training process only requires training the GNN. Therefore, GraphAdpater can efficiently pre-train based on different scales of PLMs.

Table 7.1: Statistics of experiment datasets.

Dataset	# Nodes	# Edges	# Tokens	Split ratio (%)	#Class	Metric
Arxiv	169,343	1,166,243	35,920,710	54/18/28	40	Accuracy
Instagram	11,339	144,010	579,263	10/10/80	2	ROC-AUC
Reddit	33,434	198,448	6,748,436	10/10/80	2	Accuracy

7.3.3 Fine-tuning with prompts

The pipeline is shown in Figure 7.2 (b). GraphAdapter is pre-trained by token-level semantic understanding tasks. To better utilize the learned knowledge of GraphAdapter and the PLMs in downstream tasks, we further proposed prompt-aware fine-tuning. It inserts prompts in textual data to get task-specific sentence embedding of each node. Prompts can transform various downstream tasks on TAGs into next token prediction. E.g., the task "*Which account is a student account*" can be transformed by a next-token prediction task, "[context], *based on this profile, this user is* ". In the pre-training stage, GraphAdapter has learned how to utilize the structural information captured by GNN to enhance the accuracy of next-token prediction, therefore, under the transformed downstream task can better utilize the learned knowledge from pre-training. Formally, given textual data $\mathbb{S}_i$ of node i, we can combine a sequence of tokens with task-specific prompts behind textual data, namely, $\mathbb{S}_{i|\mathbb{P}} = [\mathbb{S}_i, \mathbb{P}]$, then we can get its sentence hidden states $h_{i|\mathbb{P}}$ through the transformer of PLM. The resulting hidden state is then fused with the node's structural representation as node representation in a specific downstream task.

$$r_{i|\mathbb{P}} = \mathbf{Fusion}(h_{i|\mathbb{P}}, z_i) \tag{7.15}$$

This node representation can be used in various tasks. For example, in the node classification, we can append a new linear transformation to output the result, i.e., $\hat{y}_{i|\mathbb{P}} = f(r_{i|\mathbb{P}}|\theta_{new})$. In fine-tuning stage, the whole parameters $\{\Theta_g, \Theta_{fuse}, \theta_{new}\}$ in GraphAdapter are trainable.

7.4 EXPERIMENT

To comprehensively validate that GraphAdapter can mine the intrinsic correlation between the textual and structure data in TAGs, we conduct extensive experiments on three real-world datasets from diverse domains.

Our experiments aim to answer the following five questions:

- ***Q1*: How well is GraphAdapter in modeling TAGs?**

- ***Q2*: Whether GraphAdapter can adapt to other PLMs?**
- ***Q3*: Are all components comprising GraphAdapter valid?**
- ***Q4*: What exactly does GraphAdapter's pre-training learn?**
- ***Q5*: How efficient is GraphAdapter?**

7.4.1 Experiment setup

Dataset. We select three public and real-world datasets used for evaluation: **Ogbn-arxiv[132]**: Ogbn-Arxiv (shorted as Arxiv), is a citation network where edges represent citation relationships, nodes represent papers and the text attribute is the abstracts of papers. The task on this graph is to predict paper subjects. **Instagram[201]**: Instagram is a social network where edges represent following relationships, nodes represent users, and the prediction task is to classify commercial and normal users in this network. The text attribute is the user's profile. **Reddit**[11]: Reddit is also a social network where each node denotes a user, the node features are the content of users' historically published subreddits, and edges denote whether two users have replied to each other. The prediction task is to classify whether a user is in the top 50% popular (average score of all subreddits). Table 7.1 shows detailed statistics of these datasets.

Baselines. We compare the proposed GraphAdapter with several state-of-the-art TAG modeling methods.

- **GNN-based methods**: This method directly combines different frozen PLM with GNNs to model TAGs. Since the specific GNN framework is not the key point this paper focuses on, this paper uses GraphSAGE [10] as an instance of GNN.
- **LM-based methods**: we select GIANT [113, 117], and GLEM as baseline. GIANT use self-supervised task to finetune PLM. Then incorporates the fine-tuned PLM and GNN to model TAG. GLEM jointly trains PLM and GNN. Note, GIANT is based on BERT, and GLEM uses DeBERTa. Considering PLMs have a high influence on performance, we also compare GraphAdapter with them under the same PLM.
- **LLM-based methods**: There are a few LLM-based methods that are suitable in our setting. Therefore, we select TAPE [124] as the LLM-based baseline. This method, due

[11]`https://convokit.cornell.edu/documentation/subreddit.html`

to its need to obtain the interpretation of the text graph through GPT-3.5 and only the interpretation data on Arxiv is published. Therefore, we only report the results of this method on Arxiv.

Since many baseline methods involve GNN components, which are mostly optional, and considering that different GNNs have different performances. To make a fair comparison and without loss of generality, all GNNs used in all baselines are fixed to GraphSAGE, which is a classic and general GNN model.

Prompts. Since GraphAdapter involves prompts, to make a fair comparison, we also enhance the baselines with prompts. We provide detailed records of the prompts used in different experiments and how prompts are added to the baselines in the appendix. Meanwhile, we also validate the stability of our method with different prompts, and the experiment results can be found in the appendix. However, since prompts are not the main contribution of our method, this paper does not explore prompt methods such as soft-prompt and chain-of-thought in detail.

Table 7.2: The performance of different methods across three datasets. Each row corresponds to a specific method, and each column presents the performance of the models on a particular dataset. The evaluation metric used is accuracy for the Arxiv and Reddit datasets, and ROC-AUC for Instagram. The LM employed in each method is indicated in parentheses.

		Arxiv	**Instagram**	**Reddit**
LM	GNN (Ogb-feature)	0.6980 (0.0013)	-	-
	GNN (RoBERTa)	0.7129 (0.0013)	0.6123 (0.0063)	0.6191 (0.0043)
	GNN (RoBERTa+Prop)	0.7067 (0.0011)	0.6138 (0.0117)	0.6198 (0.0036)
	GIANT (BERT)	0.7262 (0.0011)	0.5986 (0.0022)	0.6379 (0.0045)
	GIANT (BERT+Prop)	0.7252 (0.0012)	0.6029 (0.0123)	0.6348 (0.0039)
	GLEM[1] (DeBERTa)	0.7550 (0.0024)	-	-
	GLEM (DeBERTa)	0.7355 (0.0034)	0.6166 (0.0056)	0.6228 (0.0060)
	GLEM (DeBERTa+Prop)	0.7315 (0.0033)	0.6105 (0.0038)	0.6221 (0.0052)
LLM	GNN (Llama 2)	0.7305 (0.0020)	0.6221 (0.0112)	0.6320 (0.0041)
	GNN (Llama 2+Prop)	0.7336 (0.0018)	0.6312 (0.0051)	0.6324 (0.0033)
	TAPE (GPT-3.5)	0.7672 (0.0007)	-	-
Ours	GraphAdapter (w/o Pre)	0.7648 (0.0020)	0.6351 (0.0077)	0.6284 (0.0025)
	GraphAdapter	**0.7707 (0.0015)**	**0.6513 (0.0075)**	**0.6461 (0.0019)**

[12]performance reported in [117]

7.4.2 Performance

***Q1*: How well is GraphAdapter in modeling TAGs?**

***A1*: GraphAdapter can effectively model TAGs and surpass current state-of-the-art baselines on node classification tasks.** We compare GraphAdapter with 6 state-of-the-art baselines on 3 different real-world datasets to evaluate its effectiveness. As Table 7.2 shows, the experiment results suggest:

(1) Frozen LLMs are effective on TAGs. In general, frozen LLMs have an improved performance compared to the previous frozen LM. Experiment results show Llama 2 has improved performance on 3 datasets by 1.34% compared to RoBERTa-based methods. LLM can better combine the information in prompts to extract task-relevant sentence representations of nodes. As the results show, prompts can bring a 0.42% improvement on average for LLM, but they could not improve the performance of LM. Frozen LLMs with prompt can surpass many GNN-LM methods that require tuning LM. Results also show that LLMs with prompts can surpass GLEM and GIANT by 0.43% and 0.79% on average, respectively.

(2) Directly fusing GNN and LLM results in unstable improvements. Compared to ordinary GNN, GraphAdapter (w/o Pre) only adds one fusion component to fuse the semantic representation from the LM and structural representation from the GNN. Experiment results show that directly fusing language model representations only brings improvements on Arxiv, but not obviously on other datasets. Note that the Arxiv training samples are much larger than the other datasets. This result suggests that training samples may have an impact on GNNs to understand and effectively incorporate the representations inferred by LLMs with prompts.

(3) GraphAdapter can effectively combine GNN and LLM, surpassing existing state-of-the-art baselines in terms of performance. The pre-training effect of GraphAdapter is significant, bringing an average performance improvement of 1.98% and thus surpassing existing state-of-the-art baselines. Specifically, GraphAdapter achieves an improvement of 4.72% over state-of-the-art cascaded GNN-LM methods and 5.40% over self-supervised GNN-LMs on average. At the same time, GraphAdapter also surpasses TAPE, another LLM-based method on Arxiv by 0.4% accuracy improvement.

Table 7.3: The performance of the GraphAdapter based on different LM across three datasets.

	Arxiv			**Instagram**			**Reddit**		
	RoBERTa	GPT2	Llama 2	RoBERTa	GPT-2	Llama 2	RoBERTa	GPT-2	Llama 2
GNN (PLM)	0.7129 (0.0013)	0.7174 (0.0019)	0.7305 (0.0022)	0.6123 (0.0063)	0.6019 (0.0124)	0.6221 (0.0112)	0.6191(0.0043)	0.6282 (0.0036)	0.6320 (0.0041)
GNN (PLM+Prop)	0.7067 (0.0011)	0.6915 (0.0021)	0.7336 (0.0027)	0.6138 (0.0117)	0.6128 (0.0014)	0.6312 (0.0051)	0.6198 (0.0036)	0.6206 (0.0011)	0.6324 (0.0033)
GraphAdapter (w/o Pre)	0.7069 (0.0026)	0.7146 (0.0025)	0.7648 (0.0020)	0.6165 (0.0038)	0.6162 (0.0066)	0.6351 (0.0077)	0.6210 (0.0036)	0.6284 (0.0027)	0.6369 (0.0025)
GraphAdapter	**0.7273 (0.0021)**	**0.7325 (0.0022)**	**0.7707 (0.0015)**	**0.6292 (0.0033)**	**0.6276 (0.0034)**	**0.6508 (0.0033)**	**0.6379 (0.0061)**	**0.6441 (0.0022)**	**0.6461 (0.0019)**

***Q2*: Whether GraphAdapter can adapt to other PLMs?**

Table 7.4: The performance of different methods using the same LMs across three datasets.

	Arxiv	**Instagram**	**Reddit**
GNN (BERT)	0.7039 (0.0013)	0.5973 (0.0063)	0.6061 (0.0043)
GIANT (BERT)	**0.7269 (0.0021)**	0.5986 (0.0022)	**0.6379 (0.0045)**
GraphAdapter (BERT)	0.7264 (0.0012)	**0.6156 (0.0032)**	0.6366 (0.0034)
GNN (RoBERTa)	0.7129 (0.0013)	0.6123 (0.0063)	0.6191 (0.0043)
GLEM (RoBERTa)	**0.7308 (0.0029)**	0.6114 (0.0075)	0.6228 (0.0018)
GraphAdapter (RoBERTa)	0.7273 (0.0021)	**0.6276 (0.0034)**	**0.6379 (0.0061)**

***A2*: GraphAdapter can be effectively pre-trained based on RoB-ERTa, GPT-2, and Llama 2, resulting in performance improvements.** We run GraphAapter based on 3 different LM. The experiment results are shown in Table 7.3. GraphAdapter improved average performance over directly combining GNNs with frozen PLM by 1.67% on RoBERTa, 1.89% on GPT-2, and 2.77% on Llama 2. Meanwhile, GraphAdapter pre-training brings 1.67%, 1.50%, and 1.02% improvements on RoBERTa, GPT-2, and Llama 2 respectively. This result fully demonstrates that **GraphAdapter is a general and scalable method.** It is worth noting that the pre-training method of RoBERTa is different from others. GraphAdapter uses a pre-training task similar to RoBERTa, so there are some slight differences from the formula in Section 4. The main differences come from the loss function and language model inputs. We describe the details of applying GraphAdapter on Roberta in the appendix.

Under the same PLM, the performance of GraphAdapter is comparable to the SOTA baselines based on fine-tuning the PLM. We evaluate the performance difference between GraphAdapter and SOTA baselines under the same LM. Since the GLEM adopted DeBERTa, however, the pre-training code of DeBERTa is not open-sourced at present. To keep consistent, GraphAdapter and GLEM both adopt the same RoBERTa-base. As shown in Table 7.4, the experiment results suggest that methods based on pre-training like GIANT and GraphAdapter perform better on small datasets like Instagram and Reddit. Similarly, Roberta-based GraphAdapter outperforms GLEM by 1.57% and BERT-based GIANT outperforms GLEM by 1.15% on small datasets. Compared to baselines based on pre-training, although GIANT fine-tunes the LM, its performance is 0.51% lower than GraphAdapter on average. Therefore, overall, even without fine-tuning the LM, the performance of GraphAdapter is comparable to current state-of-the-art baselines based on fine-tuning the LM.

Table 7.5: The performance of GraphAdapter when various components are removed. The evaluation metrics used for these tests align with those described above. The term 'w/o' indicates the removal of a specific component from the GraphAdapter.

	Arxiv	**Instagram**	**Reddit**
w/o Pretraining	0.7648 (0.0020)	0.6392 (0.0086)	0.6369 (0.0025)
w/o Graph structure	0.7604 (0.0024)	0.6346 (0.0074)	0.6147 (0.0012)
w/o Res label	0.7605 (0.0013)	0.6408 (0.0130)	0.6363 (0.0036)
w/o task-specific prompt	0.7594 (0.0030)	0.6364 (0.0073)	0.6430 (0.0021)
GraphAdapter	0.7707 (0.0015)	0.6513 (0.0075)	0.6461 (0.0019)

7.4.3 In-depth Analysis.

***Q3*: Are all components comprising GraphAdapter valid?**

***A3*: As Table 7.5 shows, removing any component of GraphAdapter results in performance drops.** Removing pre-training leads to an 0.91% drop, demonstrating that GraphAdapter's improvements indeed come from pre-training. Next, the most significant performance drop is when we simultaneously remove pre-training and graph structure in the fine-tuning stage (keeping only self-loops), which causes a 1.95% drop. This shows having the graph is crucial for GraphAdapter to work. Removing the task-related prompt leads to a 0.98% drop, validating our design of aligning pre-training tasks via prompts. Notably, removing the residual learning ("w/o Res Label" that is stated in section 7.3.2) leads to a 1.02% drop (more than removing pre-training), suggesting that training GNNs directly on all text may introduce excessive noise and hurt performance. Our Equation 7, which utilizes language model predictions to select words more semantically related to the graph, is reasonable.

However, although the ablation study validates the rationality of GraphAdapter's design and the efficacy of its components, these results hardly answer what exactly GraphAdapter pre-training is doing. Therefore, we further construct validation experiments about pre-training.

***Q4*: What exactly does GraphAdapter's pre-training learn?**

We conduct three comparative experiments to demonstrate what GraphAdapter pre-training is doing.

(1) GNN can obtain stronger expressive power through pre-training. We first observe the performance change of GNNs before and after pre-training, where we directly use the structural representations from the pre-trained GNN to fine-tune for downstream tasks. As Table 7.6 shows, the pre-trained GNN performs better on downstream tasks, improving

Table 7.6: The performance changes of the GNN block in GraphAdapter before and after pre-training. Here, "w/o Pretraining" signifies no pre-training, while "w Pretraining" indicates the opposite.

	Arxiv	**Instagram**	**Reddit**
GNN w/o Pretraining	0.7305 (0.0020)	0.6181 (0.0112)	0.6320 (0.0041)
GNN w Pretraining	**0.7335 (0.0024)**	**0.6294 (0.0038)**	**0.6410 (0.0027)**

by 0.78% on average. This demonstrates that GNNs are training their ability to model the graph structure during pre-training.

(2) Fusion block is learning how to fuse the knowledge from the language model and GNN during pre-training. We further explore whether the fusion layer learned useful knowledge during training. We randomly initialize the parameters in a specific GraphAdapter's blocks after pre-trained. As Table 7.7 shows, initializing the parameters of the fusion layer leads to significant performance drops, decreasing by 1.03% on average across 3 datasets. Even on the Arxiv dataset, the performance is lower than full initialization. This result shows that the enhanced knowledge from GNN may need to be outputted through the matching fusion layer. To further verify this conjecture, we further reinitialized the parameters of GNN, and some performance decline can also be observed, decreasing by 0.82% on average. This is similar to the impact of reinitializing the fusion layer. The fusion layer alone does not contain much knowledge. Therefore, these results demonstrate that the fusion layer can learn how to fuse the knowledge from GNN and language models.

(3) Graph structure is the basis of pre-training. We further observe the changes in different base models before and after pre-training. In this comparative experiment, we keep all the structures of GraphAdapter, only replacing the GNN block with MLPs of equal parameter size. As Figure 7.3 shows, the MLP-based GraphAdapter shows no significant change before and after pre-training (average improvement of 0.19%), and even decreases in performance on Instagram and Reddit (drops of 0.05% and 0.62% respectively). While the GNN improves notably before and after pre-training (average improvement of 0.91%). This result suggests that GNN is a prerequisite for effective pre-training.

These three results demonstrate that GraphAdapter is indeed learning graph structures via pre-training. This validates that language-structure pre-training of GraphAdapter is reasonable and effective, and further supports the motivation of this paper.

Table 7.7: The performance of GraphAdapter after randomly initializing some blocks. Here, "Re-init" represents re-initialization.

	Arxiv	**Instagram**	**Reddit**
Re-init All	0.7648 (0.0020)	0.6392 (0.0086)	0.6369 (0.0025)
Re-init GNN	0.7680 (0.0022)	0.6390 (0.0050)	0.6364 (0.0026)
Re-init Fusion	0.7562 (0.0011)	0.6431 (0.0024)	0.6378 (0.0022)
GraphAdapter	0.7707 (0.0015)	0.6513 (0.0075)	0.6461 (0.0019)

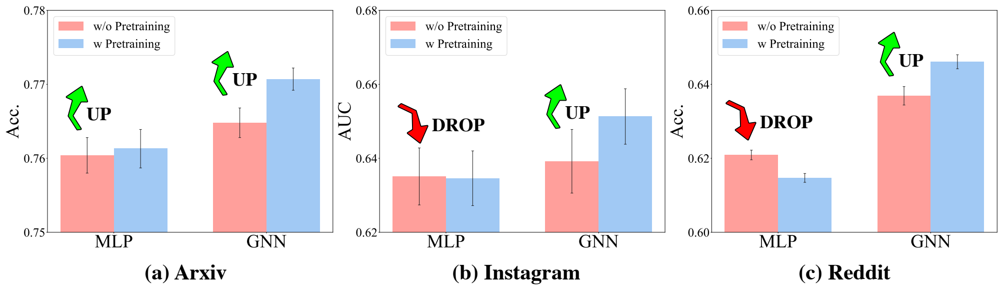


Figure 7.3: The performance of GraphAdapter before and after pre-training, using MLP and GNN as the backbone architectures. The red represents performance without pre-training, while the blue represent performance after pre-training.

7.4.4 Efficient

***Q5*: How efficient is GraphAdapter?**

As shown in Table 7.8, we demonstrate the efficiency of GraphAdapter. As can be seen in the Table, even when combined with a large model with 13B parameters, GraphAdapter has a speed comparable to PLM-based text-attributed graph modeling methods.

Table 7.8: Running time of different methods on Arxiv using one Nvidia A100 80GB. Since different methods use different PLM, we also report the number of parameters for the PLM (decoded as "# para") and the number of trainable parameters ("# trainable").

	GIANT	**GLEM**	**GraphAdapter**
PLM	BERT	DeBERTa-Large	Llama 2-13B
# para of PLM	110M	139M	13B
# trainable in Pre	110M	-	3M
# trainable in Fine	0.7M	139M	2M
Pre-process	-	-	192 min
Pre-training	341 min	-	312 min
Fine-tuning	1 min	612 min	1 min
Total time costs	342 min	612 min	505 min

CHAPTER 8: CONCLUSIONS

In this dissertation, I study *label-efficient* and *parameter-efficient* representation learning on different applications for text-rich graphs. Prior research neglects these challenges on complex text-rich graphs and exhibit limited performance on real-world networks when training data or computation budget is limited. The proposed principles and techniques help researchers to deploy their advanced machine learning methods on text-rich graphs. The future work of this thesis will delve deeper into the prospects of latent representation-based graph machine learning, emphasizing its enduring relevance across various industry applications, including retrieval and ranking.

While working on this dissertation, I recognize several notable future directions of my research and I have started to work on some of these topics.

Leverage text-rich graph learning to industrial applications. Text-rich graphs are ubiquitous in industrial applications. However, most of the deployed production model do not utilize text-rich graphs as the data model. For example, people tend to use traditional one tower or two tower neural network architecture for tasks like product or post recommendation. The current public graph benchmark datasets are typically simplistic and are different from industry datasets. This prevents academic researchers from conducting research and developing effective techniques for complex industry use cases and makes it difficult to use public graph benchmarks to evaluate these techniques. Currently, the most well-known public benchmark is Open Graph Benchmark [132]. Most of the datasets in the benchmark are homogeneous graphs with simple features (word2vec embeddings) or no features at all and each graph. As such, progress from the text-rich graph learning community cannot benefit the real-world applications. Consequently, engineers in the industry model these tasks in the traditional format. One promising direction is to set up large-scale graph learning benchmarks on text-rich graphs and categorize popular tasks into representative research problems.

Multi-task Generalization on Text-Rich Graphs. Large language models demonstrates mind-blowing unseen task generalization capabilities and paves the road to the first generation of artificial general intelligence (AGI). In contrast to text domain, it is very difficult to perform transfer learning across different tasks on graph structured data, especially those structural relevant ones. Can we utilize the generalization power of LLMs on text-rich graphs ? The *label-efficient* studies of this dissertation is the first attempt on this topic. In the future, it is important for research on text-rich graphs to yield similar level of unseen task generalization through techniques like instruction fine-tuning or Reinforcement learning from human feedback (RLHF). Additionally, the development of graph-aware large language models (LLMs) and the integration of graph tasks with text represent an interesting and impactful direction for future research.

REFERENCES

[1] Q. Zhu, N. Ponomareva, J. Han, and B. Perozzi, "Shift-robust gnns: Overcoming the limitations of localized graph training data," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 27 965–27 977.

[2] Q. Zhu, C. Zhang, C. Park, C. Yang, and J. Han, "Shift-robust node classification via graph adversarial clustering," *arXiv preprint arXiv:2203.15802*, 2022.

[3] Y. Shi, Q. Zhu, F. Guo, C. Zhang, and J. Han, "Easing embedding learning by comprehensive transcription of heterogeneous information networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2190–2199.

[4] Q. Zhu, H. Wei, B. Sisman, D. Zheng, C. Faloutsos, X. L. Dong, and J. Han, "Collective multi-type entity alignment between knowledge graphs," in *Proceedings of The Web Conference 2020*, 2020, pp. 2241–2252.

[5] Q. Zhu, Y. Jiao, N. Ponomareva, J. Han, and B. Perozzi, "Explaining and adapting graph conditional shift," *arXiv preprint arXiv:2306.03256*, 2023.

[6] Q. Zhu, C. Yang, Y. Xu, H. Wang, C. Zhang, and J. Han, "Transfer learning of graph neural networks with ego-graph information maximization," *Advances in Neural Information Processing Systems*, vol. 34, pp. 1766–1779, 2021.

[7] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[9] A. Tsitsulin, J. Palowitch, B. Perozzi, and E. Müller, "Graph clustering with graph neural networks," *arXiv preprint arXiv:2006.16904*, 2020.

[10] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.

[11] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[12] Y. Sun and J. Han, "Mining heterogeneous information networks: principles and methodologies," *Synthesis Lectures on Data Mining and Knowledge Discovery*, vol. 3, no. 2, pp. 1–159, 2012.

[13] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference.* Springer, 2018, pp. 593–607.

[14] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, 2019, pp. 2022–2032.

[15] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 793–803.

[16] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang, "Representation learning for attributed multiplex heterogeneous network," *arXiv preprint arXiv:1905.01669*, 2019.

[17] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," *Advances in neural information processing systems*, vol. 32, 2019.

[18] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *Proceedings of the web conference 2020*, 2020, pp. 2704–2710.

[19] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 992–1003, 2011.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[21] Y. Dong, Z. Hu, K. Wang, Y. Sun, and J. Tang, "Heterogeneous network representation learning." in *IJCAI*, vol. 20, 2020, pp. 4861–4867.

[22] C. Yang, Y. Xiao, Y. Zhang, Y. Sun, and J. Han, "Heterogeneous network representation learning: A unified framework with survey and benchmark," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 10, pp. 4854–4873, 2020.

[23] Q. Lv, M. Ding, Q. Liu, Y. Chen, W. Feng, S. He, C. Zhou, J. Jiang, Y. Dong, and J. Tang, "Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021, pp. 1150–1160.

[24] B. Jin, Y. Zhang, Q. Zhu, and J. Han, "Heterformer: A transformer architecture for node representation learning on heterogeneous text-rich networks," *arXiv preprint arXiv:2205.10282*, 2022.

[25] B. Jin, W. Zhang, Y. Zhang, Y. Meng, X. Zhang, Q. Zhu, and J. Han, "Patton: Language model pretraining on text-rich networks," *arXiv preprint arXiv:2305.12268*, 2023.

[26] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[27] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 1067–1077.

[28] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 855–864.

[29] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013, pp. 2787–2795.

[30] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[31] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Advances in neural information processing systems*, 2013, pp. 926–934.

[32] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," *arXiv preprint arXiv:1412.6575*, 2014.

[33] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 119–128.

[34] T. Chen and Y. Sun, "Task-guided and path-augmented heterogeneous network embedding for author identification," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 2017, pp. 295–304.

[35] J. Shang, M. Qu, J. Liu, L. M. Kaplan, J. Han, and J. Peng, "Meta-path guided embedding for similarity search in large-scale heterogeneous information networks," *arXiv preprint arXiv:1610.09769*, 2016.

[36] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2017, pp. 135–144.

[37] T.-y. Fu, W.-C. Lee, and Z. Lei, "Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 1797–1806.

[38] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous Graph Neural Network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '19*. ACM Press, 2019. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3292500.3330961 pp. 793–803.

[39] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, 2019. [Online]. Available: https://doi.org/10.1145/3308558.3313562 pp. 2022–2032.

[40] H. T. Trung, N. T. Toan, T. Van Vinh, H. T. Dat, D. C. Thang, N. Q. V. Hung, and A. Sattar, "A comparative study on network alignment techniques," *Expert Systems with Applications*, vol. 140, p. 112883, 2020.

[41] S. Zhang and H. Tong, "Attributed network alignment: Problem definitions and fast solutions," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[42] H. Zhu, R. Xie, Z. Liu, and M. Sun, "Iterative entity alignment via joint knowledge embeddings." in *IJCAI*, 2017, pp. 4258–4264.

[43] M. Chen, Y. Tian, K.-W. Chang, S. Skiena, and C. Zaniolo, "Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment," *arXiv preprint arXiv:1806.06478*, 2018.

[44] M. Chen, Y. Tian, M. Yang, and C. Zaniolo, "Multilingual knowledge graph embeddings for cross-lingual knowledge alignment," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 2017, pp. 1511–1517.

[45] Z. Sun, W. Hu, Q. Zhang, and Y. Qu, "Bootstrapping entity alignment with knowledge graph embedding." in *IJCAI*, 2018, pp. 4396–4402.

[46] Y. Cao, Z. Liu, C. Li, J. Li, and T.-S. Chua, "Multi-channel graph neural network for entity alignment," *arXiv preprint arXiv:1908.09898*, 2019.

[47] K. Xu, L. Wang, M. Yu, Y. Feng, Y. Song, Z. Wang, and D. Yu, "Cross-lingual knowledge graph alignment via graph matching neural network," *arXiv preprint arXiv:1905.11605*, 2019.

[48] Z. Wang, Q. Lv, X. Lan, and Y. Zhang, "Cross-lingual knowledge graph alignment via graph convolutional networks," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 349–357.

[49] P. Konda, S. Das, P. Suganthan GC, A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. Naughton et al., "Magellan: Toward building entity matching management systems," *Proceedings of the VLDB Endowment*, vol. 9, no. 12, pp. 1197–1208, 2016.

[50] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra, "Deep learning for entity matching: A design space exploration," in *Proceedings of the 2018 International Conference on Management of Data*. ACM, 2018, pp. 19–34.

[51] M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, and N. Tang, "Deeper–deep entity resolution," *arXiv preprint arXiv:1710.00597*, 2017.

[52] X. Chu, I. F. Ilyas, and P. Koutris, "Distributed data deduplication," *Proceedings of the VLDB Endowment*, vol. 9, no. 11, pp. 864–875, 2016.

[53] X. Dong, A. Halevy, and J. Madhavan, "Reference reconciliation in complex information spaces," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM, 2005, pp. 85–96.

[54] I. Bhattacharya and L. Getoor, "Collective entity resolution in relational data," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 5, 2007.

[55] L. Zhu, M. Ghasemi-Gol, P. Szekely, A. Galstyan, and C. A. Knoblock, "Unsupervised entity resolution on multi-type graphs," in *International semantic web conference*. Springer, 2016, pp. 649–667.

[56] J. Pujara and L. Getoor, "Generic statistical relational entity resolution in knowledge graphs," *arXiv preprint arXiv:1607.00992*, 2016.

[57] M. Pershina, M. Yakout, and K. Chakrabarti, "Holistic entity matching across knowledge graphs," in *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, 2015, pp. 1585–1590.

[58] H. Han, L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis, "Two supervised learning approaches for name disambiguation in author citations," in *Proceedings of the 2004 Joint ACM/IEEE Conference on Digital Libraries, 2004*. IEEE, 2004, pp. 296–305.

[59] H. Han, H. Zha, and C. L. Giles, "Name disambiguation in author citations using a k-way spectral clustering method," in *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, 2005, pp. 334–343.

[60] G. Louppe, H. T. Al-Natsheh, M. Susik, and E. J. Maguire, "Ethnicity sensitive author disambiguation using semi-supervised learning," in *international conference on knowledge engineering and the semantic web*. Springer, 2016, pp. 272–287.

[61] M. Song, E. H.-J. Kim, and H. J. Kim, "Exploring author name disambiguation on pubmed-scale," *Journal of informetrics*, vol. 9, no. 4, pp. 924–941, 2015.

[62] P. Treeratpituk and C. L. Giles, "Disambiguating authors in academic publications using random forests," in *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*, 2009, pp. 39–48.

[63] Y. Zhang, F. Zhang, P. Yao, and J. Tang, "Name disambiguation in aminer: Clustering, maintenance, and human in the loop." in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 1002–1011.

[64] K. Jhawar, D. K. Sanyal, S. Chattopadhyay, P. K. Bhowmick, and P. P. Das, "Author name disambiguation in pubmed using ensemble-based classification algorithms," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, 2020, pp. 469–470.

[65] S. Subramanian, D. King, D. Downey, and S. Feldman, "S2and: A benchmark and evaluation system for author name disambiguation," in *2021 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. IEEE, 2021, pp. 170–179.

[66] K. Kim, S. Rohatgi, and C. L. Giles, "Hybrid deep pairwise classification for author name disambiguation," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 2369–2372.

[67] X. Fan, J. Wang, X. Pu, L. Zhou, and B. Lv, "On graph-based name disambiguation," *Journal of Data and Information Quality (JDIQ)*, vol. 2, no. 2, pp. 1–23, 2011.

[68] J. Tang, A. C. Fong, B. Wang, and J. Zhang, "A unified probabilistic framework for name disambiguation in digital library," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 6, pp. 975–987, 2011.

[69] Z. Zhang, C. Wu, Z. Li, J. Peng, H. Wu, H. Song, S. Deng, and B. Wang, "Author name disambiguation using multiple graph attention networks," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[70] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Mach. Learn.*, vol. 79, no. 1–2, p. 151–175, May 2010. [Online]. Available: https://doi.org/10.1007/s10994-009-5152-4

[71] Y. Mansour, M. Mohri, and A. Rostamizadeh, "Domain adaptation: Learning bounds and algorithms," *CoRR*, vol. abs/0902.3430, 2009. [Online]. Available: http://arxiv.org/abs/0902.3430

[72] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[73] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, p. 97–105.

[74] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, p. 2208–2217.

[75] W. Zellinger, B. A. Moser, T. Grubinger, E. Lughofer, T. Natschläger, and S. Saminger-Platz, "Robust unsupervised domain adaptation for neural networks via moment alignment," *Information Sciences*, vol. 483, pp. 174–191, May 2019. [Online]. Available: https://doi.org/10.1016/j.ins.2019.01.025

[76] W. M. Kouw, "An introduction to domain adaptation and transfer learning," *CoRR*, vol. abs/1812.11806, 2018. [Online]. Available: http://arxiv.org/abs/1812.11806

[77] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," in *International Conference on Learning Representations*, 2019.

[78] Q. Zhu, C. Yang, Y. Xu, H. Wang, C. Zhang, and J. Han, "Transfer learning of graph neural networks with ego-graph information maximization," *Advances in Neural Information Processing Systems*, vol. 34, pp. 1766–1779, 2021.

[79] X. Ma, T. Zhang, and C. Xu, "Gcan: Graph convolutional adversarial network for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8266–8276.

[80] M. Wu, S. Pan, C. Zhou, X. Chang, and X. Zhu, "Unsupervised domain adaptive graph convolutional networks," in *Proceedings of The Web Conference 2020*, 2020.

[81] X. Shen, Q. Dai, S. Mao, F.-l. Chung, and K.-S. Choi, "Network together: Node classification via cross-network deep network embedding," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[82] J. Palowitch and B. Perozzi, "Debiasing graph representations via metadata-orthogonal training," in *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2020, pp. 435–442.

[83] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," *Advances in neural information processing systems*, vol. 14, 2001.

[84] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[85] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[86] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[87] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 385–394.

[88] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[89] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[90] J. Liu, A. Kumar, J. Ba, J. Kiros, and K. Swersky, "Graph normalizing flows," in *Advances in Neural Information Processing Systems*, 2019, pp. 13 556–13 566.

[91] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *International Conference on Learning Representations*, 2019.

[92] F.-Y. Sun, J. Hoffman, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," in *International Conference on Learning Representations*, 2019.

[93] J. Chen, T. Ma, and C. Xiao, "Fastgcn: Fast learning with graph convolutional networks via importance sampling," in *International Conference on Learning Representations*, 2018.

[94] C. Yang, A. Pal, A. Zhai, N. Pancha, J. Han, C. Rosenberg, and J. Leskovec, "Multisage: Empowering gcn with contextualized multi-embeddings on web-scale multipartite networks," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 2434–2443.

[95] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[96] L. Lan, P. Wang, X. Du, K. Song, J. Tao, and X. Guan, "Node classification on graphs with few-shot novel labels via meta transformed network embedding," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[97] J. Baek, D. B. Lee, and S. J. Hwang, "Learning to extrapolate knowledge: Transductive few-shot out-of-graph link prediction," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[98] Z. Hu, Y. Dong, K. Wang, K.-W. Chang, and Y. Sun, "Gpt-gnn: Generative pre-training of graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1857–1867.

[99] Z. Hu, C. Fan, T. Chen, K.-W. Chang, and Y. Sun, "Pre-training graph neural networks for generic structural feature extraction," *arXiv preprint arXiv:1905.13728*, 2019.

[100] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, "Gcc: Graph contrastive coding for graph neural network pre-training," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 1150–1160.

[101] R. Levie, W. Huang, L. Bucci, M. M. Bronstein, and G. Kutyniok, "Transferability of spectral graph convolutional neural networks," *arXiv preprint arXiv:1907.12972*, 2019.

[102] R. Levie, E. Isufi, and G. Kutyniok, "On the transferability of spectral graph filters," in *2019 13th International conference on Sampling Theory and Applications (SampTA)*. IEEE, 2019, pp. 1–5.

[103] L. Ruiz, L. Chamon, and A. Ribeiro, "Graphon neural networks and the transferability of graph neural networks," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[104] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[105] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, June 2018, pp. 2227–2237.

[106] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[107] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[108] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*, 2020.

[109] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever et al., "Language models are unsupervised multitask learners," *OpenAI blog*, 2019.

[110] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell et al., "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[111] G. Xun, K. Jha, J. Sun, and A. Zhang, "Correlation networks for extreme multi-label text classification," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1074–1082.

[112] Z. Liu, C. Xiong, M. Sun, and Z. Liu, "Fine-grained fact verification with kernel graph attention network," *arXiv preprint arXiv:1910.09796*, 2019.

[113] E. Chien, W.-C. Chang, C.-J. Hsieh, H.-F. Yu, J. Zhang, O. Milenkovic, and I. S. Dhillon, "Node feature extraction by self-supervised multi-scale neighborhood prediction," *arXiv preprint arXiv:2111.00064*, 2021.

[114] C. Mavromatis, V. N. Ioannidis, S. Wang, D. Zheng, S. Adeshina, J. Ma, H. Zhao, C. Faloutsos, and G. Karypis, "Train your own gnn teacher: Graph-aware distillation on textual graphs," *arXiv preprint arXiv:2304.10668*, 2023.

[115] C. Li, B. Pang, Y. Liu, H. Sun, Z. Liu, X. Xie, T. Yang, Y. Cui, L. Zhang, and Q. Zhang, "Adsgnn: Behavior-graph augmented relevance modeling in sponsored search," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 223–232.

[116] J. Yang, Z. Liu, S. Xiao, C. Li, D. Lian, S. Agrawal, A. Singh, G. Sun, and X. Xie, "Graphformers: Gnn-nested transformers for representation learning on textual graph," *Advances in Neural Information Processing Systems*, vol. 34, pp. 28 798–28 810, 2021.

[117] J. Zhao, M. Qu, C. Li, H. Yan, Q. Liu, R. Li, X. Xie, and J. Tang, "Learning on large-scale text-attributed graphs via variational inference," *arXiv preprint arXiv:2210.14709*, 2022.

[118] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale et al., "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[119] A. Zeng, X. Liu, Z. Du, Z. Wang, H. Lai, M. Ding, Z. Yang, Y. Xu, W. Zheng, X. Xia et al., "Glm-130b: An open bilingual pre-trained model," *arXiv preprint arXiv:2210.02414*, 2022.

[120] Z. Chen, H. Mao, H. Li, W. Jin, H. Wen, X. Wei, S. Wang, D. Yin, W. Fan, H. Liu et al., "Exploring the potential of large language models (llms) in learning on graphs," *arXiv preprint arXiv:2307.03393*, 2023.

[121] J. Guo, L. Du, and H. Liu, "Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking," *arXiv preprint arXiv:2305.15066*, 2023.

[122] S. Yuan and M. Färber, "Evaluating generative models for graph-to-text generation," *arXiv preprint arXiv:2307.14712*, 2023.

[123] Y. Tian, H. Song, Z. Wang, H. Wang, Z. Hu, F. Wang, N. V. Chawla, and P. Xu, "Graph neural prompting with large language models," *arXiv preprint arXiv:2309.15427*, 2023.

[124] X. He, X. Bresson, T. Laurent, and B. Hooi, "Explanations as features: Llm-based features for text-attributed graphs," *arXiv preprint arXiv:2305.19523*, 2023.

[125] K. Duan, Q. Liu, T.-S. Chua, S. Yan, W. T. Ooi, Q. Xie, and J. He, "Simteg: A frustratingly simple approach improves textual graph learning," *arXiv preprint arXiv:2308.02565*, 2023.

[126] E. B. Zaken, S. Ravfogel, and Y. Goldberg, "Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," *arXiv preprint arXiv:2106.10199*, 2021.

[127] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *International Conference on Machine Learning.* PMLR, 2019, pp. 2790–2799.

[128] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *arXiv preprint arXiv:2101.00190*, 2021.

[129] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[130] G. Penedo, Q. Malartic, D. Hesslow, R. Cojocaru, A. Cappelli, H. Alobeidli, B. Pannier, E. Almazrouei, and J. Launay, "The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only," *arXiv preprint arXiv:2306.01116*, 2023.

[131] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. V. Steeg, and A. Galstyan, "MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *Proceedings of the 36th International Conference on Machine Learning.* PMLR, 2019. [Online]. Available: http://proceedings.mlr.press/v97/abu-el-haija19a.html pp. 21–29.

[132] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *arXiv preprint arXiv:2005.00687*, 2020.

[133] Z. Liu, C. Chen, X. Yang, J. Zhou, X. Li, and L. Song, "Heterogeneous graph neural networks for malicious account detection," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 2077–2085.

[134] J. Halcrow, A. Mosoi, S. Ruth, and B. Perozzi, "Grale: Designing networks for graph learning," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2523–2532.

[135] D. Wang, J. Lin, P. Cui, Q. Jia, Z. Wang, Y. Fang, Q. Yu, J. Zhou, S. Yang, and Y. Qi, "A semi-supervised graph attentive network for financial fraud detection," in *2019 IEEE International Conference on Data Mining (ICDM).* IEEE, 2019, pp. 598–607.

[136] J. Quiñonero-Candela, M. Sugiyama, N. D. Lawrence, and A. Schwaighofer, *Dataset shift in machine learning.* Mit Press, 2009.

[137] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of statistical planning and inference*, vol. 90, no. 2, pp. 227–244, 2000.

[138] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 723–773, 2012.

[139] W. Zellinger, T. Grubinger, E. Lughofer, T. Natschläger, and S. Saminger-Platz, "Central moment discrepancy (cmd) for domain-invariant representation learning," *arXiv preprint arXiv:1702.08811*, 2017.

[140] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International conference on machine learning.* PMLR, 2019, pp. 6861–6871.

[141] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," *arXiv preprint arXiv:1810.05997*, 2018.

[142] A. Bojchevski, J. Klicpera, B. Perozzi, A. Kapoor, M. Blais, B. Rózemberczki, M. Lukasik, and S. Günnemann, "Scaling graph neural networks with approximate pagerank," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2464–2473.

[143] L.-P. Xhonneux, M. Qu, and J. Tang, "Continuous graph neural networks," in *International Conference on Machine Learning.* PMLR, 2020, pp. 10 432–10 441.

[144] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf, "Covariate shift by kernel mean matching," *Dataset shift in machine learning*, vol. 3, no. 4, p. 5, 2009.

[145] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06).* IEEE, 2006, pp. 475–486.

[146] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[147] M. Z. Y. D. H. Ren, B. L. M. C. J. Leskovec, W. Hu, and M. Fey, "Open graph benchmark: Datasets for machine learning on graphs," *arXiv preprint arXiv:2005.00687*, 2020.

[148] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[149] D. Khashabi, T. Khot, A. Sabharwal, P. Clark, O. Etzioni, and D. Roth, "Question answering via integer programming over semi-structured knowledge," *arXiv preprint arXiv:1604.06076*, 2016.

[150] H. Wang, F. Zhang, X. Xie, and M. Guo, "Dkn: Deep knowledge-aware network for news recommendation," in *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 2018, pp. 1835–1844.

[151] C. Lockard, P. Shiralkar, and X. L. Dong, "Openceres: When open information extraction meets the semi-structured web," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 3047–3056.

[152] L. Getoor and A. Machanavajjhala, "Entity resolution: theory, practice & open challenges," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 2018–2019, 2012.

[153] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: A web-scale approach to probabilistic knowledge fusion," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 601–610.

[154] B. D. Trisedya, J. Qi, and R. Zhang, "Entity alignment between knowledge graphs using attribute embeddings," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 297–304.

[155] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli, "Graph matching networks for learning the similarity of graph structured objects," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, 2019, pp. 3835–3845.

[156] F. M. Suchanek, S. Abiteboul, and P. Senellart, "Paris: Probabilistic alignment of relations, instances, and schema," *arXiv preprint arXiv:1111.7164*, 2011.

[157] P. Singla and P. Domingos, "Entity resolution with markov logic," in *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 2006, pp. 572–582.

[158] Q. Zhang, Z. Sun, W. Hu, M. Chen, L. Guo, and Y. Qu, "Multi-view knowledge graph embedding for entity alignment," *arXiv preprint arXiv:1906.02390*, 2019.

[159] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

[160] M. Wang, L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, M. Li, J. Zhou, Q. Huang, C. Ma, Z. Huang, Q. Guo, H. Zhang, H. Lin, J. Zhao, J. Li, A. J. Smola, and Z. Zhang, "Deep graph library: Towards efficient and scalable deep learning on graphs," *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. [Online]. Available: https://arxiv.org/abs/1909.01315

[161] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[162] N. Keriven and G. Peyré, "Universal invariant and equivariant graph neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 7090–7099.

[163] K. Oono and T. Suzuki, "Graph neural networks exponentially lose expressive power for node classification," in *International Conference on Learning Representations*, 2020.

[164] C. Yang, P. Zhuang, W. Shi, A. Luu, and P. Li, "Conditional structure generation through graph variational generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2019, pp. 1338–1349.

[165] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *Advances in neural information processing systems*, vol. 31, 2018.

[166] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[167] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *ACHA*, vol. 30, no. 2, pp. 129–150, 2011.

[168] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra et al., "Matching networks for one shot learning," *Advances in neural information processing systems*, vol. 29, 2016.

[169] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *International conference on learning representations*, 2016.

[170] X. Kan, H. Cui, and C. Yang, "Zero-shot scene graph relation prediction through commonsense knowledge integration," in *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21.* Springer, 2021, pp. 466–482.

[171] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, "Analysis of representations for domain adaptation," *Advances in neural information processing systems*, vol. 19, 2006.

[172] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *Journal of Machine Learning Research*, vol. 11, pp. 1201–1242, 2010.

[173] K. Borgwardt, E. Ghisu, F. Llinares-López, L. O'Bray, and B. Rieck, "Graph kernels: State-of-the-art and future challenges," *arXiv preprint arXiv:2011.03854*, 2020.

[174] N. M. Kriege, F. D. Johansson, and C. Morris, "A survey on graph kernels," *Applied Network Science*, vol. 5, no. 1, pp. 1–42, 2020.

[175] G. Nikolentzos, G. Siglidis, and M. Vazirgiannis, "Graph kernels: A survey," *arXiv preprint arXiv:1904.12218*, 2019.

[176] L. Bai and E. R. Hancock, "Fast depth-based subgraph kernels for unattributed graphs," *Pattern Recognition*, vol. 50, pp. 233–245, 2016.

[177] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," in *International Conference on Learning Representations*, 2019.

[178] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "GraphRNN: Generating realistic graphs with deep auto-regressive models," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018, pp. 5708–5717.

[179] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, and J. Huang, "Graph representation learning via graphical mutual information maximization," in *Proceedings of The Web Conference 2020*, 2020, pp. 259–270.

[180] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[181] F. R. Chung and F. C. Graham, *Spectral graph theory*. American Mathematical Soc., 1997, no. 92.

[182] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[183] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, vol. 27, no. 1, pp. 415–444, 2001.

[184] S. Verma and Z.-L. Zhang, "Stability and generalization of graph convolutional neural networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, ser. KDD '19, 2019.

[185] S. Arora, E. Hazan, and S. Kale, "Fast algorithms for approximate semidefinite programming using the multiplicative weights update method," in *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*. IEEE, 2005, pp. 339–348.

[186] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 2005, pp. 177–187.

[187] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74, no. 1, p. 47, 2002.

[188] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *International Conference on Machine Learning.* PMLR, 2020, pp. 4116–4126.

[189] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li, "Rolx: structural role extraction & mining in large graphs," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 1231–1239.

[190] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 697–706.

[191] W. Ammar, D. Groeneveld, C. Bhagavatula, I. Beltagy, M. Crawford, D. Downey, J. Dunkelberger, A. Elgohary, S. Feldman, V. Ha et al., "Construction of the literature graph in semantic scholar," *arXiv preprint arXiv:1805.02262*, 2018.

[192] Z. Lu, "Pubmed and beyond: a survey of web tools for searching biomedical literature," *Database*, vol. 2011, 2011.

[193] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J. Hsu, and K. Wang, "An overview of microsoft academic service (mas) and applications," in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 243–246.

[194] A. Kumar, A. Raghunathan, R. Jones, T. Ma, and P. Liang, "Fine-tuning can distort pretrained features and underperform out-of-distribution," *arXiv preprint arXiv:2202.10054*, 2022.

[195] A. Cohan, S. Feldman, I. Beltagy, D. Downey, and D. S. Weld, "Specter: Document-level representation learning using citation-informed transformers," *arXiv preprint arXiv:2004.07180*, 2020.

[196] X. Liu, D. Yin, X. Zhang, K. Su, K. Wu, H. Yang, and J. Tang, "Oag-bert: Pre-train heterogeneous entity-augmented academic language models," *arXiv preprint arXiv:2103.02410*, 2021.

[197] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "Mpnet: Masked and permuted pre-training for language understanding," *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 857–16 867, 2020.

[198] C. Berge, *The theory of graphs.* Courier Corporation, 2001.

[199] M. E. Newman, D. J. Watts, and S. H. Strogatz, "Random graph models of social networks," *Proceedings of the national academy of sciences*, vol. 99, no. suppl_1, pp. 2566–2572, 2002.

[200] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 974–983.

[201] S. Kim, J.-Y. Jiang, M. Nakada, J. Han, and W. Wang, "Multimodal post attentive profiling for influencer marketing," in *Proceedings of The Web Conference 2020*, 2020, pp. 2878–2884.

[202] C. D. Corley, D. J. Cook, A. R. Mikler, and K. P. Singh, "Text and structural data mining of influenza mentions in web and social media," *International journal of environmental research and public health*, vol. 7, no. 2, pp. 596–615, 2010.

[203] X. Liu, K. Ji, Y. Fu, W. L. Tam, Z. Du, Z. Yang, and J. Tang, "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks," *arXiv preprint arXiv:2110.07602*, 2021.

[204] Y. Sun, S. Wang, Y. Li, S. Feng, H. Tian, H. Wu, and H. Wang, "Ernie 2.0: A continual pre-training framework for language understanding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 05, 2020, pp. 8968–8975.

[205] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever et al., "Improving language understanding by generative pre-training," 2018.

[206] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[207] T. Gao, X. Yao, and D. Chen, "Simcse: Simple contrastive learning of sentence embeddings," *arXiv preprint arXiv:2104.08821*, 2021.

[208] T. Jiang, J. Jiao, S. Huang, Z. Zhang, D. Wang, F. Zhuang, F. Wei, H. Huang, D. Deng, and Q. Zhang, "Promptbert: Improving bert sentence embeddings with prompts," *arXiv preprint arXiv:2201.04337*, 2022.

[209] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," *arXiv preprint arXiv:2104.08691*, 2021.

[210] J. Gasteiger, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," *arXiv preprint arXiv:1810.05997*, 2018.

[211] Y. Yang and X. Cui, "Bert-enhanced text graph neural network for classification," *Entropy*, vol. 23, no. 11, p. 1536, 2021.