Towards Graph Foundation Models: A Survey and Beyond

JIAWEI LIU, CHENG YANG*, Beijing University of Posts and Telecommunications, China
ZHIYUAN LU, JUNZE CHEN, YIBO LI, Beijing University of Posts and Telecommunications, China
MENGMEI ZHANG, TING BAI, Beijing University of Posts and Telecommunications, China
YUAN FANG, Singapore Management University, Singapore
LICHAO SUN, Lehigh University, USA
PHILIP S. YU, University of Illinois Chicago, USA
CHUAN SHI†, Beijing University of Posts and Telecommunications, China

Foundation models have emerged as critical components in a variety of artificial intelligence applications, and showcase significant success in natural language processing and several other domains. Meanwhile, the field of graph machine learning is witnessing a paradigm transition from shallow methods to more sophisticated deep learning approaches. The capabilities of foundation models to generalize and adapt motivate graph machine learning researchers to discuss the potential of developing a new graph learning paradigm. This paradigm envisions models that are pre-trained on extensive graph data and can be adapted for various graph tasks. Despite this burgeoning interest, there is a noticeable lack of clear definitions and systematic analyses pertaining to this new domain. To this end, this article introduces the concept of Graph Foundation Models (GFMs), and offers an exhaustive explanation of their key characteristics and underlying technologies. We proceed to classify the existing work related to GFMs into three distinct categories, based on their dependence on graph neural networks and large language models. In addition to providing a thorough review of the current state of GFMs, this article also outlooks potential avenues for future research in this rapidly evolving domain.

CCS Concepts: • **Computing methodologies** → **Neural networks**.

Additional Key Words and Phrases: Graph Foundation Models; Large Language Models

ACM Reference Format:
Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S. Yu, and Chuan Shi. 2023. Towards Graph Foundation Models: A Survey and Beyond. 1, 1 (December 2023), 35 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Jiawei Liu and Cheng Yang contributed equally to this research.
†Chuan Shi is the corresponding author.

Authors' addresses: Jiawei Liu, Cheng Yang, {liu_jiawei,yangcheng}@bupt.edu.cn, School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, 100876, China; Zhiyuan Lu, Junze Chen, Yibo Li, {luzy,junze,yiboL}@bupt.edu.cn, School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, 100876, China; Mengmei Zhang, Ting Bai, mgwave87@gmail.com,baiting@bupt.edu.cn, Beijing University of Posts and Telecommunications, Beijing, 100876, China; Yuan Fang, yfang@smu.edu.sg, School of Computing and Information Systems, Singapore Management University, Singapore, 188065, Singapore; Lichao Sun, lis221@lehigh.edu, School of Computing and Information Systems, Lehigh University, Pennsylvania, 18015, USA; Philip S. Yu, psyu@uic.edu, University of Illinois Chicago, Chicago, 60607, USA; Chuan Shi, shichuan@bupt.edu.cn, School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, 100876, China.

XXXX-XXXX/2023/12-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

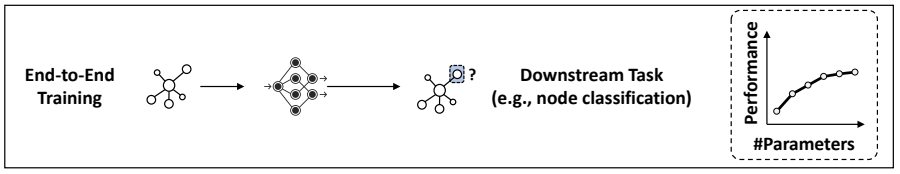
1 INTRODUCTION

With the rise in computational power and breakthroughs in deep learning techniques, the artificial intelligence (AI) community has introduced the notion of "foundation models": *A foundation model is any model that is trained on broad data and can be adapted to a wide range of downstream tasks [2].* Foundation models enjoy unique attributes like emergence and homogenization, empowering them to serve as the primary building blocks for a myriad of downstream AI applications [2]. Emergence suggests that as a foundation model scales up, it may spontaneously manifest novel capabilities [133]. Meanwhile, homogenization alludes to the model's versatility, enabling its deployment across diverse applications [2]. Thanks to the development of large language models (LLMs), the concept of foundation models first became a reality in natural language processing (NLP). Since then, foundation models have demonstrated impressive versatility, processing not just text but also image data, video data, audio data and multi-modal inputs. This versatility empowers them to excel in tasks ranging from computer vision [128] and audio signal processing [164] to recommender systems [20].

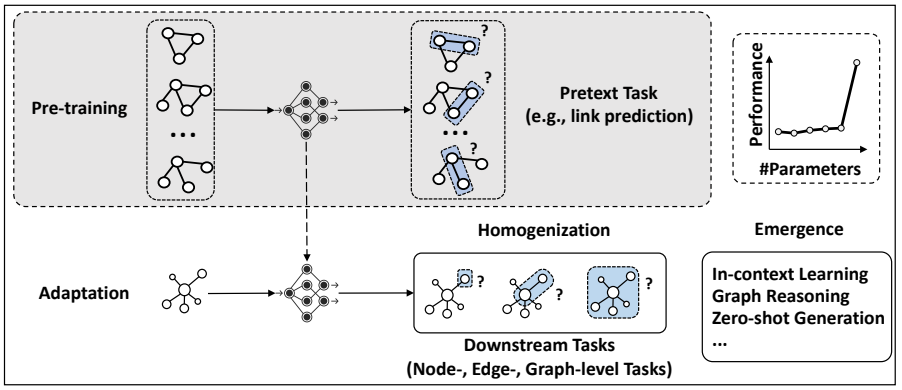
Much like the evolution witnessed in NLP, graph machine learning is also undergoing a paradigm transition. In its early stages, graph tasks predominantly employed shallow methods, such as random walk [88] and matrix factorization [150]. These methods, however, were typically limited to transductive learning on unattributed graphs [122]. The more recent shift towards deep learning methods has catalyzed the rise of graph neural networks (GNNs). GNNs have revolutionized the landscape by introducing the message-passing mechanism, where nodes iteratively aggregate information from their neighbors. By harnessing GNNs in fully supervised, semi-supervised, or unsupervised settings, researchers have pioneered a variety of customized graph models. These advancements have yielded substantial improvements in tasks like node classification [57], link prediction [168], graph classification [149], and graph clustering [125]. However, certain challenges of GNN models still persist. For example, GNNs are restricted with issues related to expressive power [1] and generalizability [154], especially given the ever-expanding datasets and the widening spectrum of tasks.

The remarkable success of foundation models in varied domains is increasingly garnering the interest of graph machine learning researchers. This naturally evokes the question: Could graph foundation models represent the next frontier in graph machine learning? Such models, if realized, would boast enhanced expressive power, improved transferability, and applicability to more intricate graph data and tasks. As illustrated in Figure 1, a graph foundation model (GFM) is envisioned as a model pre-trained on extensive graph data, primed for adaptation across diverse downstream graph tasks. Drawing parallels with traditional foundation models, a GFM is also anticipated to embody two principal characteristics: emergence and homogenization. Specifically, emergence refers to novel capabilities shown exclusively in large-scale graph models, while homogenization denotes the model's adaptability across different types of graph tasks. Existing deep graph learning methods struggle to encompass these features: their inherent architectures and learning paradigms focus on specific tasks, which restrict the utilization of extensive unlabeled data, subsequently limiting their expressive and generalization abilities.

Inspired by the success of LLMs as foundation models in NLP, researchers have explored the possibilities of graph foundation models towards the emergence and homogenization capabilities. These explorations primarily focus on the design of backbone architectures for GFMs, and different learning paradigms including pre-training and adaptation, as they are the key strategies of LLMs to acheive the aforementioned capabilities. First and foremost, the emergent abilities of foundation models typically exist only in backbones with a large number of parameters, whereas the parameter count of graph neural networks is significantly smaller than that of the language backbones. This implies that the backbone of graph foundation models may need to be redesigned to achieve more substantial knowledge storage towards emergence. As graph data is typically associated with rich



(a) Deep Graph Learning.



(b) Graph Foundation Models.

Fig. 1. The distinction between deep graph learning and graph foundation models. Deep learning tackles specific tasks on specific datasets through end-to-end training. In contrast, GFMs are pre-trained on broad graph data and can be adapted to a wide range of downstream graph tasks, expected to demonstrate emergence and homogenization capabilities.

text information, an alternative approach is to use LLMs as graph foundation models. Nonetheless, it remains uncertain whether LLMs can effectively handle graph data and associated tasks, and it is crucial to determine how to model graph structures in LLMs. Additionally, the homogenization of foundation models necessitates the handling of diverse tasks in a uniform manner. Devising effective pretext tasks and downstream task adaptation methods are challenging for graph data, due to the complexity in interconnected nodes and various forms of attributes, as well as the diversity in tasks across node-, edge- and graph-levels. Therefore, there is also a need to design suitable pre-training tasks and adaptation mechanisms.

While there is no definitive solution for designing and implementing graph foundation models, this paper surveys some related researches and categorizes them into three distinct approaches based on their reliance on GNNs and LLMs: (1) **GNN-based Models**: They aim to enhance existing graph learning paradigms through innovations in the backbone, pre-training, and adaptation aspects; (2) **LLM-based Models**: They explore the feasibility of using an LLM as a graph foundation model by converting graphs into text or tokens; (3) **GNN+LLM-based Models**: They explore various forms of synergy between GNNs and LLMs to empower them with enhanced capabilities.

To the best of our knowledge, this is the first survey towards graph foundation models. Existing surveys of foundation models typically explore modalities such as language and vision [2, 177], rather than graphs. Additionally, there are two surveys [86, 87] dedicated to knowledge graphs and large language models, but knowledge graphs, due to their distinct nature in construction and application, fall outside the scope of this article. We have also noticed a very recent article that mentions the concept of large graph models [172], but it emphasizes opinion statements and lacks a systematic taxonomy. Therefore, the contributions of this article can be summarized as follows:

- This article defines the concept of graph foundation models for the first time, and examines the core issues and characteristics of their capabilities.
- This article introduces a novel taxonomy and discusses the strengths and limitations of each approach towards graph foundation models.
- This article provides promising future directions towards graph foundation models.

The subsequent sections of this article are organized as follows. In Section 2, we introduce the background related to graph foundation models. Section 3 defines graph foundation models and highlights their similarities and differences with language foundation models. Sections 4 - 6 delve into the relevant works that consider GNN-based models, LLM-based models and GNN+LLM-based models as graph foundation models, separately. Section 7 engages in a discussion on the future directions of graph foundation models. In Section 8, we summarize the key points of this paper.

2 BACKGROUND

Before introducing the concept of graph foundation models, in this section, we first review some background knowledge, namely, deep graph learning and language foundation models. Specifically, we introduce them from three aspects: data, backbone architectures, and learning paradigms.

2.1 Deep Graph Learning

Many real-world systems naturally find their representations in the form of graphs. Deep graph learning holds significant importance across various fields for its powerfulness in modeling and representing complex relationships and interactions among entities. In this section, we provide a concise overview that covers the main steps of deep graph learning, which consists of three key aspects: graph data, backbone architectures and learning paradigms.

2.1.1 Graph Data. Graph is a versatile and powerful data representation that captures intricate relationships and dependencies among entities in a network. The graph data has the following several characteristics. **(1) Non-Euclidean Nature**: Graph data is inherently non-Euclidean, as it lacks the rigid geometric structure found in traditional data formats [62]. In contrast to Euclidean data with fixed neighborhood region and a deterministic order, graph data explicitly encode complex connections between entities. **(2) Various Domains**: Graph data is ubiquitous across various domains, including social networks [24], biology [82] and transportation [47], etc. Graph data in various domains can exhibit distinct characteristics, including different node types, edge semantics, and structural patterns. For example, in biological networks, nodes can represent proteins, genes, or metabolites, while edges may indicate interactions such as protein-protein interactions or metabolic reactions. This domain-specific variability makes it challenging to create a one-size-fits-all model that can effectively generalize and adapt to diverse graph structures [42]. **(3) Various Types:** Graph data comes in various types, including homogeneous graphs [146], heterogeneous graphs [104], hypergraphs [23] and dynamic graphs [105], etc. Homogeneous graphs contain nodes and edges of the same type, such as a citation network composed of papers. Heterogeneous graphs include more than one type of node or edge, such as a citation network containing both authors and papers. Hypergraphs are composed of hyperedges connecting multiple nodes and can model high-order relationships among nodes. Dynamic graphs refer to graph structures where nodes and edges change over time, as seen in a traffic network formed by changing traffic flow.

2.1.2 Backbone Architectures. As the current mainstream backbone architecture, graph neural networks (GNNs) have emerged as a powerful framework for deep graph learning. Most GNNs follow the message-passing framework [26], which enables nodes in a graph to exchange information with their neighbors. For example, GCN [57] introduces the concept of graph convolution layers and

lays the foundation for many subsequent GNN architectures. GraphSAGE [32] proposes a method for generating embeddings for nodes in large graphs using inductive learning. Additionally, GAT [120] introduces the attention mechanism to GNNs, allowing nodes to weigh the importance of their neighbors during message passing, enhancing their expressive power. These works have significantly contributed to the advancement of GNNs, making them versatile tools for deep graph learning.

Despite the fact that deeper neural networks can achieve stronger expressive power [33], deepening GNNs is not easy. The reason is that as the number of layers in GNN increases, too much noise is introduced into the message aggregation process, causing the representations of all nodes to become similar [65]. This is also known as the over-smoothing problem. Furthermore, an increase in the number of layers leads to an exponential growth in the receptive field, giving rise to the over-squashing problem [1], where a substantial amount of information is compressed into fixed-length node vectors. In recent years, several efforts have been made to address the over-smoothing problem and over-squashing problem in deep graph neural networks, resulting in improved downstream task performance. For example, innovations like DropEdge [100] enhance GCN models by randomly removing edges for improved performance and scalability (up to 64 layers). Another thread of work to improve the expressive power of GNNs is graph transformer [7, 58, 158]. Thanks to its fully connected attention mechanism and the long-range relationship modeling capability, the graph transformer architecture can alleviate the over-smoothing problem and over-squashing problem [96].

2.1.3 Learning Paradigms. The learning paradigms for deep graph learning encompass three primary categories: supervised learning, semi-supervised learning and unsupervised learning. In this section, we will provide a brief introduction to these learning paradigms.

Supervised learning. In the supervised setting, algorithms leverage a training dataset comprising input data paired with corresponding output labels. This paradigm finds practical applications in tasks such as graph classification [59] and graph regression [48]. For example, in the molecular property prediction task [138], GNN models are trained to predict specific chemical properties or attributes of molecules using labeled training data, enabling the discovery of valuable insights for drug development and materials research.

Semi-supervised Learning. Semi-supervised learning, as highlighted in a recent study [106], constitutes the primary focus of deep graph learning. This approach harnesses both labeled and unlabeled data to enhance model performance, with node classification [57] emerging as a prominent application. The message-passing mechanism [26] empowers GNNs to iteratively exchange information among neighboring nodes. This capability enables the propagation of information throughout the graph, effectively incorporating both labeled and unlabeled data to facilitate predictions. Furthermore, GNNs can also be combined with traditional methods like label propagation to further enhance their performance in semi-supervised settings [151].

Unsupervised Learning. Unsupervised learning [95] is a broader machine learning approach to learn patterns and structures from data without manual labels. For example, graph clustering [102] aims to discover inherent structures and patterns within the graph solely based on the relationships and connectivity between nodes. Another example is link prediction, which aims to predict missing or upcoming connection relationships. An important subclass of unsupervised learning is self-supervised learning, which aims to generate labels using information inherent in the data itself [76]. Based on self-supervised learning, GNNs can be trained end-to-end and applied to downstream tasks such as graph clustering [125] and link prediction [168].

2.2 Language Foundation Models

AI is currently undergoing a transformative shift marked by the emergence of some specific natural language models (such as GPT-3) that are trained on extensive and diverse datasets using large-scale

self-supervised learning. These models, known as foundation models, are able to produce a wide array of outputs, enabling them to tackle a broad spectrum of downstream tasks. In contrast to the deep graph learning pipeline, the foundation model's approach embraces a pre-training and adaptation framework, enabling the model to achieve several significant advancements, including the emergence [133] and homogenization [2]. Foundation models have primarily established themselves in the field of NLP [2], so our discussion will focus on language foundation models in this section.

2.2.1 Language Data. Language data refers to text or spoken content in a human language, encompassing the grammar rules of the natural language and the associated semantics of the words. It can include written documents, transcribed audio recordings, and any other form of language-based communication. Language data is essential for many NLP tasks, such as machine translation, sentiment analysis and text summarization. Researchers and developers use language data to train and evaluate language models and other NLP algorithms. The quality and quantity of language data play a crucial role in the performance of NLP systems, impacting their accuracy, robustness, and overall effectiveness in various language tasks. In contrast to computer vision and other domains, the size of annotated language data is rather small, consisting of only a few thousand sentences [85]. This limitation is primarily due to the high cost of manual annotation. Nevertheless, there is a vast amount of unlabeled language data available from sources such as the internet, newspapers, and books, creating opportunities for utilizing unlabeled data in model pre-training. Furthermore, compared to graph data, language data as Euclidean data is easier to model, and its rich semantic information significantly enhances the knowledge transferability of language models.

2.2.2 Backbone Architectures. An early breakthrough in foundation models is pre-trained language models (PLMs), designed to capture context-aware word representations, which proved remarkably effective as versatile semantic features. For instance, BERT [13], grounded in the parallelizable Transformer architecture [119] with self-attention mechanisms, is conceived through the pre-training of bidirectional language models with specifically designed pretext tasks on vast unlabeled data. This landmark study significantly elevates the performance benchmarks for NLP tasks and serves as a catalyst for a plethora of subsequent research, establishing the prevailing pre-training and fine-tuning learning paradigm.

Furthermore, researchers have observed that increasing the scale of PLMs, whether by augmenting model size or training data, frequently results in increased model capacity for downstream tasks. These larger PLMs, collectively referred to as LLMs, exhibit distinctive behaviors compared to their smaller counterparts (e.g., the 1.5B-parameter GPT-2 and 175B-parameter GPT-3). After training on massive text datasets, they manifest remarkable capabilities, often referred to as emergent abilities [133], such as in-context learning [2]. LLMs primarily utilize the Transformer architecture, because highly parallelizable Transformer-based architectures accelerate the pre-training stage and enable the utilization of massive datasets. In the context of Transformer models, tokens serve as the input and represent units at the word level in natural language texts. Typically, LLMs containing hundreds of billions (or more) of parameters [176], exemplified by models such as GPT-3 [5], PaLM [12], Galactica [116], and LLaMA [118].

2.2.3 Learning Paradigms. As the number of model parameters has rapidly increased, the demand for significantly larger datasets has grown to effectively train these parameters and avoid overfitting. Given the extremely expensive costs associated with building large-scale labeled datasets, the importance of utilizing extensive unlabeled text data has been underscored. Leveraging these unlabeled datasets involves a two-step approach: first, achieving universal representations through self-supervised learning, and subsequently employing these representations for various tasks [93].

Based on different adaptation approaches, learning paradigms can be categorized into two types: pre-train and fine-tune and pre-train, prompt, and predict [72].

Pre-train and Fine-tune. In this paradigm, a model with a consistent architecture is initially pre-trained as a language model (LM), where it predicts the probability of observed textual data. In comparison to end-to-end training, pre-training offers distinct advantages and forms cornerstone for the capabilities of foundation models. Firstly, pre-training on huge text corpus enables the learning of universal language representations, which is a possible reason to explain emergent abilities [93]. Additionally, pre-training provides improved model initialization, typically resulting in enhanced generalization performance, enabling homogenization of multiple tasks [93]. Furthermore, pre-training serves as a form of regularization, helping to prevent overfitting on smaller datasets [19]. For instance, models like GPT-3 [5] are trained with a language modeling objective, rewarding them for their abilities in emergence and homogenization.

Following the pre-training phase, foundation models acquire general-purpose capabilities suitable for a broad spectrum of tasks. Nevertheless, pre-trained models still lack downstream task-specific information, and using them directly may not yield optimal results. Therefore, we need to tune the model for specific tasks, which is known as fine-tuning. Building upon the success of models like ULMFit [40] and BERT [13], fine-tuning has emerged as the predominant method for adapting pre-trained models. In this framework, the primary emphasis lies in objective engineering, encompassing the design of training objectives for both pre-training and fine-tuning phases. For instance, Pegasus [167] shows that incorporating a loss function for predicting important sentences within a document results in an improved pre-trained model for text summarization. The advantage of fine-tuning is that it can transfer knowledge between source and target tasks (or domains) and benefit the model's performance. For the small size of fine-tuning dataset compared to pre-training dataset, this process can enable adaptation effectively without losing the stored structural language knowledge.

Pre-train, Prompt and Predict. In this paradigm, rather than adjusting pre-trained language models to suit specific downstream tasks, the approach involves reshaping the downstream tasks to align more closely with those tackled during the original LM training, accomplished by providing textual prompts. By selecting suitable prompts, we can steer the LM's behavior so that it can predict the desired output, sometimes without any additional task-specific training. This method offers the advantage of enabling a single, entirely unsupervised LM, when equipped with a set of fitting prompts, to handle a wide array of tasks [72].

From the aspect of prompt engineering, the approaches to create a proper prompts can be classified to manual methods and automated methods. Manual methods involve creating intuitive templates based on human insight, which is the most straightforward approach to crafting prompts. For instance, the influential LAMA dataset [89] offers manually devised cloze templates to assess the knowledge of language models. However, manual methods face challenges in terms of high cost and precision. To address these issues, some approaches have started to experiment with automated prompt generation. For example, Prompt Mining [52] is a template discovery approach that autonomously identifies templates based on a given set of training inputs and outputs.

Looking at it from a different perspective, in terms of how models and prompts are combined to generate results, prompting strategies can be categorized into three approaches: tuning-free prompting, prompt tuning, and instruction tuning [20]. Tuning-free prompting directly generates answers based solely on a prompt without altering the parameters of the pre-trained LLMs [118] [84]. Prompt tuning introduces supplementary prompt-relevant parameters in addition to the parameters of the pre-trained models and update these additional parameters using supervision signals acquired from downstream training samples [68] [31]. Instruction tuning adapts the LM's parameters in a manner similar to the fine-tuning process, while additionally incorporating fixed instructions to guide the model's behavior. This approach offers potential enhancements, especially in zero-shot scenarios [143].

3 GRAPH FOUNDATION MODELS

In this section, we will first formally define the concepts of graph foundation models, including the definition, key characteristics and key technologies. Then, we will discuss the impact from graph data and graph tasks on graph foundation models. Finally, we will discuss the similarities and differences between graph foundation models and language foundation models.

3.1 Concepts of Graph Foundation Model

In this subsection, we will first provide a definition of graph foundation models. Following that, we will delve into the key characteristics and essential techniques of graph foundation models, as well as the impact of graph data and graph tasks on graph foundation models.

3.1.1 Definition and Key Characteristics. We define a graph foundation model as follows:

Definition 3.1. A graph foundation model (GFM) is a model that is expected to benefit from the pre-training of broad graph data, and can be adapted to a wide range of downstream graph tasks.

Compared to deep graph learning that employs end-to-end training, GFMs use pre-training to obtain the knowledge from a substantial amount of unlabeled graph data, and then use adaptation techniques to tailor to various downstream tasks. Some studies [78, 113] have already demonstrated that the paradigm of pre-training and adaptation outperform deep graph learning in certain scenarios, e.g., few-shot learning [78], showcasing their superior expressive power and generalization ability. Unlike deep graph learning that aims to achieve better performance on a single task, a GFM is expected to have two key characteristics: emergence and homogenization.

Emergence. Emergence means that the graph foundation model will exhibit some new abilities when having a large parameters or trained on more data, which are also referred to as emergent abilities. Inspired by the various emergent abilities [14, 63, 134] possessed by foundation models, we expect GFMs to have similar abilities, including in-context learning, graph reasoning, and zero-shot graph generation, etc. In-context learning allows predictions for various downstream tasks with few examples [46]. Graph reasoning decomposes a complex problem into multiple sub-problems based on the graph structure and addresses them step by step, such as solving graph algorithm problems [126]. Zero-shot graph generation requires the model to generate graphs based on the desired conditions without the need for any examples [108]. Note that although language foundation models have demonstrated various emergent abilities, only a few works [46, 108, 126] have explored emergent abilities of GFMs so far.

Homogenization. Homogenization means that the graph foundation model can be applied to different formats of tasks, such as node classification, link prediction and graph classification. Note that due to the distinct characteristics of tasks on graphs compared to NLP tasks, achieving homogenization is not straightforward. The fundamental question in achieving homogenization is to decide which form to unify different types of graph tasks. Existing works have attempted homogenization through link prediction [78] or graph-level tasks [113], but there is no consensus on which approach is superior.

3.1.2 Key Technologies. Graph foundation models primarily comprise two key techniques: pre-training and adaptation. This section will provide a brief overview of these two techniques.

Pre-training. Pre-training is a pivotal concept in the development of graph foundation models, akin to its role in language models. It involves pre-training a neural network on a large graph dataset in a self-supervised manner. During pre-training, the model learns to capture the structural information, relationships, and patterns within the graph. There are several pre-training strategies for graph foundation models. Contrastive self-supervised learning [110, 181] leverages the idea of learning representations by contrasting positive samples (e.g., similar node pairs) against negative

samples (e.g., dissimilar node pairs). Generative self-supervised learning [37, 38] encourages the model to reconstruct the structure or predict the features of original graph data. If using LLM as a part of the graph foundation model, we can also employ the pre-training methods introduced in Section 2.2.3. These diverse pre-training approaches enable graph foundation models to learn meaningful representations from raw graph data, enhancing their generalization and adaptability across various graph tasks.

Adaptation. The adaptation of graph foundation models involves tailoring these models to specific downstream tasks or domains to enhance their performance. This process includes several techniques, i.e., vanilla fine-tuning, parameter-efficient fine-tuning and prompt-tuning. Vanilla fine-tuning (Vanilla FT) entails training the entire pre-trained model on task-specific data, allowing for the highest level of customization but often requiring substantial data and computational resources. Parameter-efficient fine-tuning (Parameter-efficient FT) [28, 66], on the other hand, adjusts only a subset of the model's parameters, striking a balance between task-specific adaptation and resource efficiency. Prompt-tuning [111, 113] is a versatile approach that relies on external prompts to guide the model's behavior, making it more adaptable and effective. These adaptation techniques enable graph foundation models to excel in a wide range of applications by leveraging their pre-trained knowledge while tailoring their capabilities to specific tasks or domains, making them valuable for diverse downstream applications. Note that although LLMs have developed various types of prompt-tuning methods [72] and some other efficient tuning methods, such as Prefix Tuning [68], there are relatively few prompt tuning methods for graph foundation models.

3.1.3 Impact from Graph Data. The success of foundation models depends on high-quality training data, and foundation models exhibit significantly different performance on various types of test data. In this section, we discuss the impact of graph data on graph foundation models from three aspects: graph type, graph scale and graph diversity.

Graph Type. Based on the number of categories of nodes and edges in a graph, we can categorize graphs into homogeneous graphs and heterogeneous graphs. In homogeneous graphs, all nodes and edges belong to the same category. For example, in a social graph where nodes represent individuals (users) and edges represent friendship relationships, it is a homogeneous graph because all nodes are individuals and all edges represent friendship relationships. Heterogeneous graphs, on the other hand, have more than one type of nodes or edges, representing different types of entities and relationships [104]. For instance, an e-commerce graph may include nodes for users, products, and purchase relationships, forming a heterogeneous graph. For graph foundation models, handling heterogeneous graphs poses greater challenges and typically requires the design of specific backbone architectures and optimization objectives. Nonetheless, utilizing the meta-path based approach [129], a heterogeneous graph can be mapped into multiple homogeneous graphs, one for each meta-path. For example, one can apply the GFM trained on homogeneous graphs to each of these meta-path induced homogeneous graphs, separately, to get node embedding. Then, these embeddings on homogeneous graphs under different meta-paths can be fused together. However, beyond homogeneous graphs and heterogeneous graphs, there are some more complex types of graphs in the real world, such as dynamic graphs and hypergraphs [117], which poses additional challenges for GFM.

Graph Scale. Based on the number of nodes and edges in a graph, we can categorize graphs into relatively small graphs and large graphs. Small graphs are of smaller scale, typically containing dozens to hundreds of nodes and edges. For example, chemical molecular graphs represent the structure of small molecules and typically consist of dozens to hundreds of atoms. Large graphs, on the other hand, refer to graphs with a significant number of nodes and edges, often encompassing millions or even billions of nodes and edges. For instance, e-commerce graph in Alibaba includes billons of nodes and hundreds of billion edges [163]. For graph foundation models, large graphs impose higher

demands on the capacities of graph foundation models. Firstly, large graphs, due to their numerous nodes and typically sparser edges, introduce more noise and pose greater challenges in terms of storage and computation [152]. Additionally, large graphs often exhibit long-range dependency relationships [17], requiring more neural network layers and a higher number of parameters, which exacerbates the over-smoothing [65] and over-squashing [1] problem of GNN-based models.

Graph Diversity. Based on whether a graph dataset originates from the same domain, we can categorize graphs into same-domain graphs and cross-domain graphs. Same-domain graphs refer to graph data from similar or related domains, typically containing nodes and edges of similar types. For example, the social graphs of Facebook and WeChat come from similar domains. Cross-domain graphs [179], on the other hand, involve graph data from different domains or data sources, often comprising nodes and edges of different types, aimed at addressing multi-domain problems or cross-domain tasks. For example, academic networks and molecular graphs come from different domains. For graph foundation models, supporting cross-domain graphs presents greater challenges because graphs from different domains lack a unified underlying semantics. This can result in weak transfer performance or even negative transfer when applying the model to a new dataset [178]. Therefore, addressing the heterogeneity of different domains and enabling the same GFM to be applied to graphs from different domains is a significant challenge for GFMs.

Table 1. The relationship between language foundation model and graph foundation model.

		Language Foundation Model	Graph Foundation Model
Similarities	Goal	Enhancing the model's expressive power and its generalization across various tasks	
	Paradigm	Pre-training and Adaptation	
Intrinsic differences	Data	Euclidean data (text)	Non-Euclidean data (graphs) or a mixture of Euclidean (e.g., graph attributes) and non-Euclidean data
	Task	Many tasks, similar formats	Limited number of tasks, diverse formats
Extrinsic differences	Backbone Architectures	Mostly based on Transformer	No unified architecture
	Homogenization	Easy to homogenize	Difficult to homogenize
	Domain Generalization	Strong generalization capability	Weak generalization across datasets
	Emergence	Has demonstrated emergent abilities	No/unclear emergent abilities as of the time of writing

3.1.4 Impact from Graph Tasks. Language foundation models can be widely applied to various NLP tasks, while for graph foundation models, the formats of graph tasks are also quite diverse and can be categorized into three classes: node-level tasks, edge-level tasks, and graph-level tasks.

Node-level Tasks. Node-level tasks refer to the classification, regression, or prediction performed on each node. Common node-level tasks include node classification, node regression, and clustering coefficient prediction. For example, in social networks, graph nodes can represent users, and node classification can be used to identify users from different social circles or with varying interests.

Edge-level Tasks. Edge-level tasks involve the classification, regression, or prediction performed on each individual edge. Common edge-level tasks include edge classification, link prediction, shortest path prediction, connectivity prediction, and maximum flow prediction. For example, in e-commerce, link prediction can be used to predict products that users may be interested in.

Graph-level Tasks. Graph-level tasks focus on the entire graph. Common graph-level tasks include graph classification, graph regression, graph generation, graph clustering, graph condensation and average clustering coefficient prediction. For example, in bioinformatics, graph property prediction can be used to predict the biological activity or toxicity of molecular compounds, thereby accelerating the drug discovery process.

In summary, the format of tasks in graphs are highly diverse and can be categorized into three types: node-level, edge-level, and graph-level, each of which has wide-ranging applications. This undoubtedly increases the challenge of homogenization for graph foundation models. For example, in graph classification and node classification tasks on synthetic datasets, modeling structural information is often more crucial [159]. On the other hand, when dealing with node classification tasks on graphs with rich node features, modeling feature information becomes more important [159]. Furthermore, tasks that are more similar to each other will also have a lower transfer difficulty, implying that these tasks are more likely to be addressed using the same GFM. While increasing expressive power holds promise for improving the performance of many node-level, edge-level, and graph-level tasks [64], there is also some work suggesting that overly strong expressive power may not be necessary for graph generation tasks [183].

3.2 Comparison with Language Foundation Model

Through conceptual comparison, we can observe similarities in the goals and learning paradigms between graph foundation models (GFMs) and language foundation models (commonly referred to as large language models, LLMs). However, the uniqueness of graph data and graph tasks creates fundamental differences between them, which we refer to as their intrinsic differences. Furthermore, due to the relatively limited research on GFMs at present, many issues that have been extensively explored in LLMs remain unresolved, which we refer to as their extrinsic differences. We summarize the similarities and differences between GFMs and LLMs in Table 1, and will delve into them in detail in this section.

3.2.1 Similarities. As shown in Table 1, both language foundation models and graph foundation models share the common goal of enhancing a model's expressive power and improving its ability to generalize across a wide range of tasks. They aim to create versatile, pre-trained models that can be adapted for specific applications. In addition, both follow the pre-training and adaptation paradigm. They begin by pre-training a model on a large, diverse dataset and then adapt it to task-specific data.

3.2.2 Intrinsic Differences. The intrinsic differences between GFM and LLM primarily manifest in two aspects: data and tasks. As for input data, language foundation models are primarily designed for processing Euclidean data, i.e., text. They are trained on vast text corpora, which are inherently sequential and follow a linear order of words or tokens. Graph foundation models, on the other hand, are designed to handle non-Euclidean data (represented as graph structures) or a mixture of Euclidean data (like graph attributes) and non-Euclidean data. Compared to text data, graph data can capture complex data relationships and is typically more sparse. Moreover, as mentioned in Section 3.1.3, different graphs may exhibit significant differences in type or scale, all of which pose challenges in the design of GFMs. Furthermore, language data, even when sourced from texts in different domains, still share a common vocabulary. On the other hand, different graph data may lack this common foundation. For instance, nodes represent atoms in a molecular graph, while nodes represent users in a social network graph, which are entirely different.

In addition, LLMs are typically designed to handle dozens of tasks [61], but these tasks can all be unified under the format of masked language modeling. The reason is that these tasks all involve processing textual data and using the syntax and semantic information within the text. In contrast, GFMs focus a narrower set of tasks but with diverse formats. They excel at graph tasks such as node classification, link prediction and graph classification. The differences in tasks imply that GFMs cannot be learned using methods similar to those in LLMs, significantly increasing the adaptability challenges of GFMs in downstream tasks.

3.2.3 Extrinsic Differences. In addition to the intrinsic differences in data and tasks, there are also some extrinsic differences between GFMs and LLMs, which are due to the lag in technological advancements in GFMs. This section summarizes these differences as follows:

Backbone Architectures. Language foundation models, such as GPT-3 [5] and LLaMA [118], are mostly based on the Transformer architecture. The advantages of Transformer in terms of expressive power, scalability, parallelizability, and its excellent performance in handling various NLP tasks have made it the mainstream backbone architecture for LLMs. However, for GFMs, using mainstream GNNs as the backbone architecture may not necessarily be suitable. This is mainly because the expressive power and generalization of GNNs have limitations, and their parameter sizes are often too small to exhibit emergent abilities. Despite recent research efforts to design Graph Transformers [110] or models that incorporate LLMs [16], there is still no unified backbone architecture for GFMs.

Homogenization. Language foundation models are relatively easy to homogenize. This means that various NLP tasks can be formulated as the same task [94], making it possible to use a single model with unified training paradigm for a wide range of tasks. However, due to the poor transferability of graph structural knowledge, homogenization is more challenging for GFMs. Existing work attempts to achieve homogenization by unifying various tasks as link prediction [78] or graph-level tasks [113]. Additionally, constructing a data-task heterogeneous graph [46] may establish connections between different tasks, but it is a more complex process.

Domain Generalization. Language foundation models have demonstrated strong domain generalization capabilities. They can often perform well on tasks and datasets that were not seen during training, showcasing their ability to generalize across various language-related domains. However, due to the diversity and lack of unified vocabulary of graph data, GFMs generally exhibit weaker generalization across datasets, especially when moving to cross-domain graph data [178]. Their performance may degrade significantly when faced with graph structures or characteristics that differ substantially from their training data. Achieving strong domain generalization remains a challenging research problem for GFMs.

Emergence. Language foundation models have shown emergent abilities, where they can generate coherent and contextually relevant text based on few examples or instructions. Representative emergent abilities include in-context learning [14], chain of thought reasoning [134] and zero-shot generation [63]. However, GFMs have not demonstrated obvious emergent abilities to the same extent as language foundation models. Only a few recent studies discuss the in-context learning [46], graph reasoning [126] and zero-shot graph generation [108] abilities of GFMs.

3.3 Summary

In this section, we define the concept of graph foundation models and related technologies, and compares graph foundation models with language foundation models. In the following sections, we will introduce three categories of methods for implementing graph foundation models, along with representative works for each method, as illustrated in Figure 2. GNN-based models use GNN as the backbone architecture, while LLM-based models transform the graph into the input format of LLM and use LLM as the backbone architecture. GNN+LLM-based models, on the other hand, utilize both GNN and LLM as the backbone architecture simultaneously. The distinction in backbone architecture also impacts the methods for pre-training and adaptation. Therefore, in the following sections, we will introduce the backbone architectures, pre-training, and adaptation strategies for each category of methods, seperately.

4 GNN-BASED MODELS

Thanks to effective model architectures and training paradigms, language models have achieved remarkable performance in natural language processing tasks. The backbone, pre-training and

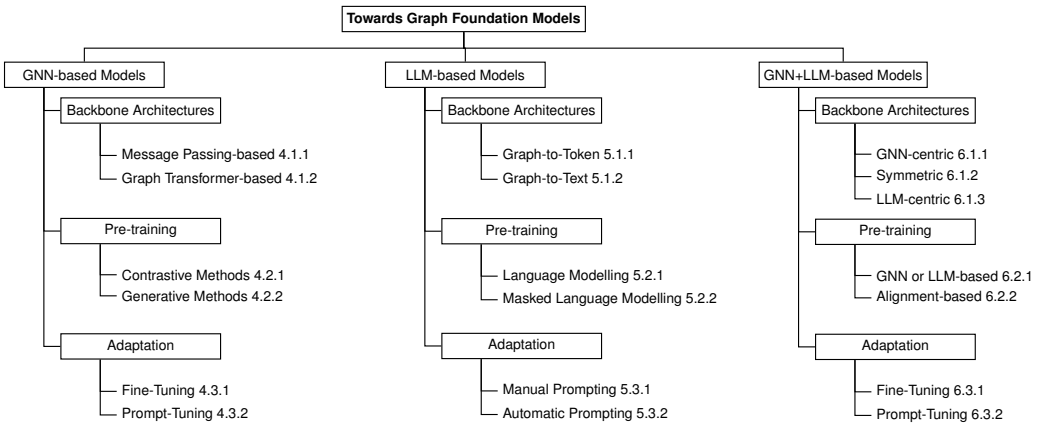
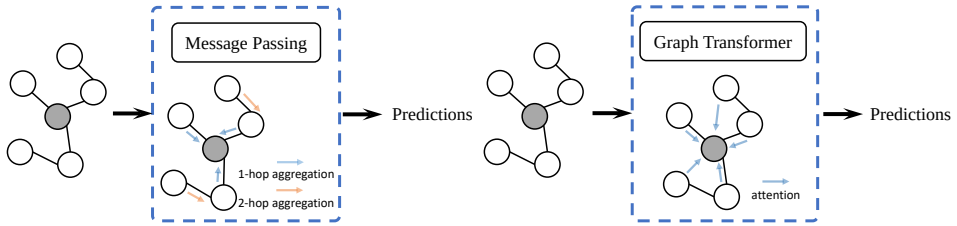


Fig. 2. Taxonomy of existing works towards graph foundation models.



(a) Message Passing. (b) Graph Transformer.

Fig. 3. A comparison between message passing-based models and graph transformer. A fundamental distinction is that the message passing mechanism is constrained by the graph structure, and the graph transformer treats the graph as a fully-connected network.

adaptation techniques employed in language models have inspired a series of corresponding efforts in the field of graph-based tasks. In this section, we will delve into GNN-based models, which draw inspiration from the model architectures or training paradigms used in NLP to apply them to graph tasks. Importantly, unlike the LLM-based models and GNN+LLM-based models to be introduced in the following sections, GNN-based models do not explicitly model text data in their pipeline. We have summarized and categorized the works mentioned in this section in Table 2.

4.1 Backbone Architectures

Numerous GNNs have been proposed and they have been widely used in various graph-related downstream tasks. These networks are leveraged for feature extraction, often serving as the foundational components of graph models, commonly referred to as "backbones". In this subsection, we introduce two advanced GNN backbones: message-passing-based and transformer-based methods.

4.1.1 Message Passing-Based Methods. Message Passing Neural Networks (MPNNs) [26] represent a broad category of GNN architectures that operate based on the concept of message passing between nodes in a graph. In the message passing mechanism, each node aggregates information from its neighboring nodes, processes the information, and then sends messages to its neighbors in a series of iterative steps. A typical message passing process can be formulated as :

$$h_v^{k+1} = U_k(h_v^k, \sum_{u \in N(v)} M_k(h_v^k, h_u^k, \mathbf{X}_{(u,v)}^e)), \tag{1}$$

Table 2. Details of approaches involved as GNN-based models.

Model	Backbone Architecture	Pre-training	Adaptation
All In One [113]	GCN, GAT, Graph Transformer	Same-Scale CL	Prompt Tuning
PRODIGY [46]	GCN, GAT	Graph Reconstruction, Supervised	Prompt Tuning
DGI [121]	GCN	Cross-Scale CL	Parameter-Efficient FT
GRACE [181]	GCN	Same-Scale CL	Vanilla FT
VGAE [56]	GCN	Graph Reconstruction, Property Prediction	Vanilla FT
MA-GCL [27]	GCN	Same-Scale CL	Vanilla FT
GraphMAE [38]	GAT	Graph Reconstruction	Parameter-Efficient FT
GraphMAE2 [37]	GAT	Graph Reconstruction	Parameter-Efficient FT
GPPT [111]	GraphSAGE	Graph Reconstruction, Cross-Scale CL	Prompt Tuning
VPGNN [137]	GraphSAGE	Cross-Scale CL	Prompt Tuning
GPT-GNN [44]	HGT	Graph Reconstruction	Vanilla FT
PT-HGNN [49]	HGT	Same-Scale CL	Vanilla FT
CPT-HG [50]	HGT	Same-Scale CL	Vanilla FT
GraphPrompt [78]	GIN	Graph Reconstruction	Prompt Tuning
GCC [92]	GIN	Same-Scale CL	Vanilla FT
GraphCL [160]	GIN	Same-Scale CL	Parameter-Efficient FT
AdapterGNN [66]	GIN	Cross-Scale CL, Graph Reconstruction, Same-Scale CL	Parameter-Efficient FT
AAGOD [30]	GIN	Same-Scale CL, Supervised	Prompt Tuning
GPF [21]	GIN	Cross-Scale CL, Graph Reconstruction	Prompt Tuning
SGL-PT [180]	GIN	Same-Scale CL, Graph Reconstruction	Prompt Tuning
Graph-BERT [166]	Graph Transformer	Graph Reconstruction, Supervised	Vanilla FT
GROVER [99]	Graph Transformer	Property Prediction	Vanilla FT
G-Adapter [28]	Graph Transformer	Supervised, Graph Reconstruction, Property Prediction	Parameter-Efficient FT

where h_v^k and h_v^{k+1} denote the embedding of node v at layer k and layer $k+1$, $\mathbf{X}^e_{(u,v)}$ denotes the edge attribute of edge (u, v), $N(v)$ denotes the neighbors of node v, $M_k(\cdot)$ and $U_k(\cdot)$ denote the message and update function at layer k.

Many existing GNN-based models utilize message passing-based models as their backbone. Due to the simplicity and effectiveness, several studies [27, 46, 56, 113, 121, 181] adopt GCN [57] as their backbone architecture, where GCN [57] employs a localized first-order approximation of spectral graph convolutions for the dual purpose of capturing graph structure and encoding node features. Several studies [37, 38, 46, 113] adopt GAT [120] as their backbone architecture, where GAT [120] replaces the average aggregation operation in GCN with a weighted aggregation approach, facilitated by an attention mechanism. Additionally, a multi-head attention technique can be further used in GAT to enhance its performance. GPPT [111] and VPGNN[137] uses GraphSAGE [32] as their backbone, which operates by sampling a fixed-size subset of neighboring nodes for each target node and then learns embeddings by aggregating and processing these sampled neighbors' embeddings. For heterogeneous graphs, some works [44, 49, 50] use HGT [45] as their backbone, which introduces a specialized attention mechanism for neighborhood aggregation. Unlike global attention, HGT employs type-specific parameters to define heterogeneous attention over each edge within the graph. To improve the expressive power, a proportion of studies rely on GIN [149] as their primary architecture [21, 30, 66, 78, 92, 160, 180]. GIN is a message passing-based model with expressive power theoretically equivalent to a 1-WL test [135]. Due to the expressive power of GIN, it is frequently chosen as the backbone for many GNN-based graph models. For an in-depth exploration of message passing-based GNNs, we recommend referring to [144].

4.1.2 Graph Transformer-Based Methods. While GNNs have demonstrated significant success in learning from graph data, they still confront fundamental limitations, including issues related to limited expressive power [149], over-smoothing [65] and over-squashing [1]. In parallel, the transformer architecture [119], which has revolutionized tasks in natural language processing

[5, 13] and computer vision [15, 77], achieving the state-of-the-art performance. It has inspired the development of transformer-based models tailored for graph data [7, 55, 58, 158]. Graph transformers have exhibited promising results, particularly in molecular prediction tasks [158], owing to their fully-connected self-attention mechanism. This mechanism enables them to address the shortcomings of traditional message-passing GNNs thanks to its long-range modeling capability and strong expressive power.

The principal distinction between the backbone architectures with message passing mechanism and the graph transformer lies in their treatment of the underlying graph structure. In the case of the graph transformer, it treats the graph as if it were fully connected, meaning it considers and measures the similarity between every pair of nodes in the graph. Conversely, the message passing mechanism operates under the constraint of the adjacency matrix of the graph. It only propagates information between nodes that are explicitly connected in the graph structure. We illustrate the difference between message passing-based models and graph transformers in Figure 3.

Currently, there are many research efforts focusing on graph transformers. Here we will present part of these studies that employ a pre-training and fine-tuning learning paradigm: Graph-BERT [166] uses intimacy-based and hop-based relative positional embeddings to encode node positions in a subgraph. The intimacy-based relative positional embeddings capture the relative positions of nodes in a subgraph based on their connectivity patterns. The hop-based relative distance embeddings capture the relative positions of nodes in a subgraph based on their hop distances. GROVER [166] uses a variant of MPNN called Directed Message Passing Networks (DyMPNs), which can capture the directed nature of molecular graphs and distinguish different types of edges. The DyMPNs in GROVER are used to compute the node and edge embeddings in the Transformer-style architecture. Graphormer [158] introduces a spatial encoding method to represent structural node relationships. It assigns learnable embeddings to node pairs based on their spatial proximity, derived from the shortest path distance. This distance is incorporated as a bias term in softmax attention to improve spatial dependency capture in graphs. For a more comprehensive exploration, please refer to other literature on graph transformers [80, 81].

4.2 Pre-training

Pre-training in the field of NLP involves exposing a model to a vast amount of unlabeled text data, allowing it to learn general language semantic knowledge in a self-supervised manner. This pre-training step equips the model with a foundational understanding of language, enabling it to transfer this knowledge to downstream tasks. Similarly, the graph domain typically includes many unlabeled nodes and graphs, providing opportunities for pre-training on graphs. Graph pre-training enables the graph models to understand graph structure and semantic information, thus encoding meaningful node or graph embeddings [142, 148]. Recently, some graph pre-training methods have been proposed to learn representations in a self-supervised manner. Based on self-supervised tasks, these graph pre-training methods can be categorized into two types: contrastive pre-training and generative pre-training.

4.2.1 Contrastive Methods. Specifically, the contrastive graph pre-training methods aim to maximize mutual information between different views, which forces the model to capture invariant semantic information across various views. The graph view can vary in scale, encompassing local, contextual, or global perspectives [142]. These perspectives correspond to node-level, subgraph-level, or graph-level information within the graph, leading to two distinct categories: (1) Same-scale contrastive learning and (2) Cross-scale contrastive learning. Same-scale contrastive learning compares two graph views at the same level. For example, GCC [92] treats the embedding of the node's subgraph as the node embedding, considering different subgraphs of the same node as positive

examples and different nodes as negative examples. Afterward, the model employs NCE loss to pull the positive samples close and push the negative samples away, which forces the encoder to capture general patterns. GraphCL [160] and GRACE [181] generate two views by graph augmentation and then employ the InfoNCE loss to contrast node-level embeddings, pushing the graph model to acquire the invariant representations. MA-GCL [27] focuses on manipulating the neural architectures of view encoders instead of perturbing graph inputs or model parameters. CPT-HG [50] proposes two pre-training tasks. The relation-level pre-training task encodes the relational semantics which constitute the basis of heterogeneity on a heterogeneous graph, while the subgraph-level pre-training task encodes high-order semantic contexts. PT-HGNN [49] considers both node- and schema-level pre-training tasks. The node-level pre-training task utilizes node relations to encourage the GNN to capture heterogeneous semantic properties, while the schema-level pre-training task utilizes the network schema to encourage the GNN to capture heterogeneous structural properties. Cross-scale contrastive learning compares two graph views at different levels. For example, DGI [121] utilizes a discriminator to maximize the mutual information between the node embeddings and the graph embedding and minimize the information between node and corrupted graph embedding. Such a contrastive process encourages the encoder to capture information of the whole graph. Although DGI enables the model to capture semantic information about nodes and graphs, it ignores the discrepancies between different nodes.

4.2.2 Generative Methods. In addition to contrastive methods, some generative graph pre-training approaches have been proposed. The aim of generative pre-training methods is to enable GNNs to understand the general structural and attribute semantics of graphs. Thus, the GNNs can be adapted to downstream tasks with universal information. Generative learning frameworks for graphs can be classified into two categories based on how they acquire generative targets [148]: Graph Reconstruction and Property Prediction.

Graph reconstruction aims to reconstruct specific parts of given graphs, emphasizing fidelity in reproducing the original graph structure or node attributes. For example, VGAE [56] extends the VAE to the graph domain, where it first employs GCN as an encoder to generate node embeddings and then reconstructs the adjacency matrix by the inner product of node embeddings. Although these methods enable models to capture the relationships between nodes and neighbors, they overlook the high-order structural information and attribute semantic information in the graph. Furthermore, GPT-GNN [44] proposes the self-supervised edge and attribute generation tasks to push the model to understand the inherent dependencies of attribute and structure. As a result, the model can learn high-order structure and semantic information. GraphMAE [38] and GraphMAE2 [37] consider that previous generative methods overemphasize structure information, instead, they employ the reconstruction of features and a re-mask decoding strategy in a self-supervised manner. In the property prediction category, models focus on learning and predicting non-trivial properties of provided graphs. For instance, GROVER [99] introduces tasks for nodes and edges, predicting context-aware properties within local subgraphs. The graph-level self-supervision task aims to predict motifs, framing it as a multi-label classification problem with each motif as a label.

Although generative methods are capable of generating novel content, the quality and interpretability of the content are hard to guarantee. In the future, it remains to be explored how to enhance the accuracy and rationality of the generative methods.

4.3 Adaptation

Typically, the objectives of pre-training tasks are different from the downstream ones, which hinders the transferability of pre-training models. To this end, fine-tuning is a common adaptation approach based on subtle adjustments of model parameters. In addition, the *"pre-train, prompt and predict"*

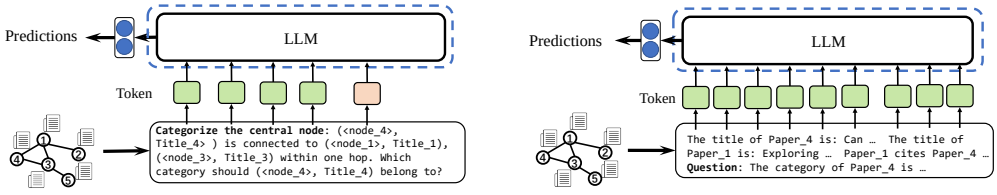
paradigm has attracted considerable attention in recent years. By using prompts, the format of downstream tasks is aligned with that of pre-training tasks, enabling pre-training models to handle downstream tasks in a more effective manner.

4.3.1 Fine-Tuning. For the situation where the model conducts the pre-training and downstream tasks in the same domain, we can utilize a pre-training model to generate node embeddings or graph embeddings, and subsequently fine-tune an external task-specific layer to generalize the pre-training model to downstream tasks. DGI [121] and GRACE [181] utilize the pre-trained encoder to obtain node embeddings, and then fine-tune a logistic regression classifier with labeled data to handle the node classification task. Additionally, there is a more practical scenario where pre-training is performed on the known graphs while tested on unseen graphs. Pre-training models cannot encode unknown graphs appropriately, thus we need to fine-tune the model in this situation. GPT-GNN [44] employs the labeled data to fine-tune a downstream task-specific decoder, which guides the pre-training model to adapt to downstream tasks. Moreover, some parameter-efficient fine-tuning methods have been proposed recently. AdapterGNN [66] employs two parallel adapters before and after the message passing stage to modify the input graph. Such addition-based methods only need to fine-tune the introduced parameters. G-Adapter [28] proposes a parameter-efficient fine-tuning method for graph transformer, which introduces graph structure into the fine-tuning by message passing. Although the fine-tuning methods have achieved significant success, they typically require sufficient labeled data to tune the model parameters. Moreover, conventional fine-tuning methods necessitate repetitive fine-tuning for each task, incurring significant computational costs. Therefore, more advanced fine-tuning techniques for graph foundation models are still to be explored.

4.3.2 Prompt Tuning. Prompt tuning has recently emerged as a strategy to circumvent the need for full-parameter tuning, facilitating both multi-task adaptation and zero-shot learning. Some studies [21, 30, 46, 78, 111, 180] focus on the prompt tuning with graph data. For example, GPPT [111] employs a prompting function to generate a token pair for each class, transforming all node classification tasks into link prediction tasks. GraphPrompt [78] unifies pre-training and downstream tasks into a consistent task format based on subgraph similarity, and utilizes labeled data to learn a task-specific prompt vector for each task, which modifies the model's Readout operation and narrows the gap between link prediction and downstream tasks. AAGOD [30] proposes to implement the data-centric manipulation by superimposing an amplifier on the adjacency matrix of the original input graph as learnable instance-specific prompts. All In One [113] converts the node-level and graph-level tasks to graph-level tasks. It treats an extra subgraph as a prompt and merges it with the node subgraph. The model subsequently utilizes the combined graph to generate predictions. GPF [21] introduces a uniform feature modification vector to each node in the graph, which can be optimized to adapt pre-training GNN models under any pre-training strategy. SGL-PT [180] incorporates a strong and general pre-training task that harnesses the advantages of both generative and contrastive approaches. Additionally, it features verbalizer-free prompting function, thus aligning the downstream task with the pre-training method's format. PRODIGY [46] is a framework for pre-training an in-context learner over prompt graphs. The goal is to enable a pretrained model to adapt to diverse downstream tasks without optimizing any parameters. Although these methods have improved the performance in few-shot scenarios, further exploration is needed to understand the semantics and interpretability of the graph prompts.

4.4 Discussion

GNN-based models offer advantages, primarily in their small parameter size, resulting in low-cost training. Additionally, GNN-based models possess essential properties like permutation invariance and often exhibit strong performance, especially in scenarios without textual attributes.



(a) Graph-to-token. (b) Graph-to-text.

Fig. 4. An illustration of two approaches to align graph data with natural language. One approach tokenizes graph data and use node representations (depicted as red tokens in the figure) as well as text tokens (depicted as green tokens in the figure) to be the input of LLMs. Another approach represents graph data with prompts in natural language and uses text tokens only (depicted as green tokens in the figure) as the input of LLMs.

However, due to the small parameter size, these models typically have limited capacity to harness extensive knowledge and can struggle to manifest emergent abilities. Moreover, due to their inherent absence of explicit text modeling, GNN-based models underutilize the information stored in pre-training language models, leading to a suboptimal exploitation of textual data. Subsequently, in the following sections, we will explore graph learning models that incorporate LLMs.

5 LLM-BASED MODELS

Researchers are actively exploring ways to leverage LLMs as core and sole backbone for graph learning [29, 126, 157], for the following advantages that can not be underestimated. Firstly, transformer-based models show a remarkable ability to seamlessly integrate textual information in graph data [157]. Additionally, employing a LLM-liked backbone empowers models to unify diverse graph learning tasks, as these tasks can be described using natural language. Furthermore, recent advancements, such as NLGraph [126], GPT4Graph [29], showcase the LLMs' prowess in preliminary graph reasoning. These advantages mark a highly promising direction for the development of such models. To discover the potential of engaging LLMs into graph learning, these works involve both graph-based properties and textual information as input for the backbone networks. Following some surveys [9, 87], our characterization of the backbone is not confined solely to the narrow definition of LLMs (like GPT-3); it also encompasses certain transformer-based models that leverage textual information. We have summarized and categorized the works mentioned in this section in Table 3.

Table 3. Details of approaches involved as LLM-based models.

Model	Backbone Architecture			Pre-training	Adaptation
InstructGLM[157]	Graph-to-token	+	Flan-T5/LLaMA	MLM,LM	Manual Prompt Tuning
LLMtoGraph[71]	Graph-to-text	+	GPTs, Vicuna	LM	Manual Prompt Tuning
NLGraph[126]	Graph-to-text	+	GPTs	LM	Manual Prompt Tuning
GraphText[175]	Graph-to-text	+	GPTs	LM	Manual Prompt Tuning
LLM4Mol[91]	Graph-to-text	+	GPTs	LM	Manual Prompt Tuning
GPT4Graph[29]	Graph-to-text	+	GPT-3	LM	Manual Prompt Tuning + Automatic Prompt Tuning
Graph-LLM[9]	Graph-to-text	+	BERT, DeBERTa, Sentence-BERT, GPTs, LLaMA	MLM,LM	Manual Prompt Tuning + Automatic Prompt Tuning

5.1 Backbone Architectures

A central question in employing LLMs for graph data is how to align graph data with natural language so that LLMs can understand them. Given that LLMs initially accept tokens as their inputs and rely on self-attention layers to process input sequences for producing hidden representations, it can be a difficult task to attain a well comprehension of graph structure information [157].

As illustrated in Figure 4, there are primarily two approaches that have been developed to address this crucial question, namely graph-to-token and graph-to-text.

5.1.1 Graph-to-token. One approach entails the tokenization of graph information and imitates the standardized input format of transformer-based models. This methodology necessitates not only the serialization of graph data into tokens but also the solutions for encoding the graph's structural information. Since this method uses node representations as unique tokens for the input of backbone models, the backbone need to be either trainable transformers or open source LLMs. For instance, InstructGLM [157] uses LLaMA [118] and T5 [94] to be their backbones for further tuning.

The concept of graph-to-token initially surfaces in GIMLET [173] that treats node representations as tokens and aims to integrate graph data with textual data. Specifically, GIMLET expands the capabilities of LLMs to accommodate both graph and text data by using the transformer architecture, incorporating generalized position embedding and instruction-based pre-training. Furthermore, efforts have been made to integrate graph data with other modalities beyond just text data. For instance, Meta-Transformer [170] introduces a transformer-based architecture designed to incorporate various forms of multimodal data, including graphs, text, and images. However, despite the promising trend indicated by developing unified multimodal intelligence using a transformer backbone, their approaches cannot be considered as graph foundation models because they do not involve any pre-training and adaptation learning paradigm.

InstructGLM [157], on the other hand, adopts a pre-training and adaptation framework and introduces LLMs to further enhance the model's text processing capabilities, making it a strong contender for the position of a graph foundation model. In this framework, the vocabulary of the LLMs is expanded by incorporating the inherent node feature vectors from the graph as distinct and unique tokens for each node. Leveraging LLMs and the transferability of natural language, InstructGLM makes a valuable contribution to the ongoing movement towards graph foundation model architectures and pipelines that span multiple modalities.

These efforts tokenize graph data to align it with natural language, enabling joint understanding with data from other modalities. Their conclusions showcase promising results for integrating graph data with natural language. However, despite these promising results, how to inform LLMs of underlying graph structures remains an important challenge in this approach.

5.1.2 Graph-to-text. To align graph data with natural language, another approach involves describing graph information using natural language. Several approaches have been developed along this line of thoughts, utilizing prompts to integrate the capabilities of large language models into classical tasks on graphs. For this method, which exclusively relies on natural language prompts, the associated backbone model can be any LLM, even if it is not open-sourced. For instance, LLMtoGraph [71] uses several LLMs including GPT-4 [83] and Vicuna [10], GPT4Graph [29] uses InstructGPT-3 [84], LLM4Mol [91] uses GPT-4 [83], Graph-LLM [9] utilizes multiple language models of different sizes, including BERT [13], DeBERTa [34], Sentence-BERT [97], GPT-4 [83] and LLaMA [118].

Initial attempts mainly use edge list to describe graph structures in natural language and make assessment on various graph tasks. LLMtoGraph [71] employs node and edge lists as the input format for graph data, conducting a series of experiments to evaluate the capacity of several LLMs to process graph data. Though their work reveals that LLMs are capable of delivering high-quality responses to problems pertaining to graph data, the prompt engineering in this stage such as zero-shot chain-of-thought or few-shot prompting may not yield the anticipated benefits. NLGraph [126] also conducts a comprehensive assessment of LLMs across eight graph reasoning tasks as well as popular GNN tasks in natural language. Similarly, utilizing edge lists to describe graph structure, the results once again underscores the limitations of this approach when dealing with complex graph problems. GraphText [175] introduces a novel approach called "Graph-syntax Tree" to describe graph data

using natural language. This approach enables graph reasoning within a text-based context. One of the notable advantages of this method is that it allows models to readily incorporate the inductive bias of GNNs by the graph-syntax tree.

Moreover, GPT4Graph [29] introduces a novel approach to prompt engineering that combines manually crafted prompts with prompts generated by the language model itself, referred to as handcrafted prompts and automatic prompts. Specifically, for manual prompting, it utilizes description languages such as edge lists, adjacency lists, Graph Modeling Language (GML), and Graph Markup Language (GraphML) to represent graph structures and compare their effectiveness. For automatic prompting, it employs techniques like graph summarization, neighborhood summarization, graph exploration, and graph completion to actively engage LLMs in understanding and manipulating graphs, facilitating graph-based reasoning and learning. The findings indicate that self-prompting is a more effective method for informing LLMs about graph structures. Graph-LLM [9] further supports this conclusion, emphasizing that neighbor summarization is the most effective technique in existing prompt engineering methods.

Furthermore, efforts have been undertaken to explore the potential of LLMs in addressing tasks from other domains using textual prompts. For instance, LLM4Mol [91] uses SMILES (Simplified Molecular Input Line Entry System) to directly describe the property of molecule, and explores how LLMs can contribute to molecular property prediction.

These studies highlight significant potential for using natural language to describe graph data and using textual tokens as the input of LLMs for graph learning. Nevertheless, a key takeaway from their conclusions is that, at the present moment, the way we use these prompts may not be an effective approach for mining underlying graph structures.

5.2 Pre-Training

The methods discussed in this section solely employ LLMs as the backbone. Hence, the pre-training phase for these methods corresponds to the pre-training phase of LLMs. There are mainly two tasks used in LLM-based models for graph learning, we will provide a concise overview of these two pre-training tasks.

5.2.1 Language Modeling (LM). Language Modeling (LM) is one of the most common self-supervised task in NLP, and is widely adopted by many state-of-the-art LLMs, such as LLaMA [118] and GPT-3 [84]. LM task can be essentially addressed to the challenge of estimating probability distributions of the next word. While LM represents a broad concept, it frequently pertains specifically to auto-regressive LM or unidirectional LM in practical applications [93]. Many methods involve LM as their pre-training method, namely InstructGLM [157], LLMtoGraph [71], NLGraph [126], GPT4Graph [29], Graph-LLM [9], GraphText [175] and LLM4Mol [91].

In the context of a text sequence represented as $s_{1:L} = [s_1, s_2, \cdots, s_L]$, its overall joint probability, denoted as $p(s_{1:L})$, can be expressed as a product of conditional probabilities, as shown in equation:

$$p(s_{1:L}) = \prod_{l=1}^{L} p(s_l | s_{0:l-1}) . \tag{2}$$

Here, s_0 represents a distinct token signifying the commencement of the sequence. The conditional probability $p(s_l|s_{0:l-1})$ is essentially a probability distribution over the vocabulary based on the linguistic context $s_{0:l-1}$. To model the context $s_{0:l-1}$, a neural encoder $f_{nenc}(\cdot)$ is employed, and the conditional probability is calculated as follows:

$$p(s_l|s_{0:l-1}) = f_{lm}(f_{nenc}(s_{0:l-1})). \tag{3}$$

In this equation, f_{lm} represents the prediction layer. By training the network using maximum likelihood estimation (MLE) with a large corpus, we can effectively learn these probabilities. Nevertheless,

a drawback of unidirectional language models is their encoding of contextual information for each token, which is solely based on the preceding leftward context tokens and the token itself. However, for more robust contextual representations of text, it is preferable to capture contextual information from both the forward and backward directions.

5.2.2 Masked Language Modeling (MLM). Masked language modeling (MLM) is introduced to address the limitation of the traditional unidirectional language model, frequently denoted as a Cloze task [93]. In MLM, specific tokens within the input sentences are randomly masked, and the model is then tasked with predicting these masked tokens by analyzing the contextual information in the surrounding text. As an effective pre-training task, MLM is adopted in BERT [13] and T5 [94]. Additionally, MLM can be categorized into two subtypes: Sequence-to-Sequence MLM (Seq2Seq MLM) and Enhanced MLM (E-MLM). Seq2Seq MLM involves utilizing the masked sequences as input for a neural encoder, and the resulting output embeddings are used to predict the masked token through a softmax classifier. On the other hand, E-MLM extends the mask prediction task to various types of language modeling tasks or enhances MLM by integrating external knowledge. MLM also faces some drawbacks as this pre-training method may result in a disconnection between the pre-training and fine-tuning stages since the mask token is absent during fine-tuning. InstructGLM [157] and Graph-LLM [9] use T5/BERT as backbones, and adopt MLM pre-training strategy.

There are also many pre-training tasks like Permuted Language Modeling (PLM) [155], Denoising Autoencoder (DAE) [60], Text-Text Contrastive Learning (TTCL), [101] and others. These pre-training tasks are currently not adopted by existing LLM-based models in graph learning, and thus not within the scope of our discussion in this section. However, we believe that in the future, more research will be developed on graph tasks involving these pre-training tasks, offering additional possibilities for the establishment and refinement of graph foundation models.

5.3 Adaptation

The adaptation phase plays a pivotal role in enhancing the performance of LLM-based models in graph learning. LLMs are primarily trained on textual corpora, which results in a significant gap between the pre-training phase and their deployment on graph tasks. Both the graph-to-token and graph-to-text methods are accompanied by specific adaptation techniques designed to enhance the LLM's ability to understand graph data effectively. As these methods share a fundamentally similar training procedure that utilizes prompts, we classify these adaptation strategies in the aspect of prompt engineering: manual and automatic.

5.3.1 Manual Prompting. Methods mentioned here use manually created prefix style prompts. For instance, LLMtoGrpah [71] and NLGraph [126] employ node and edge lists incorporating other graph properties described in natural language to form a comprehensive prompt. GPT4Graph [29] goes a step further by utilizing additional description languages to represent graph data, such as edge list, adjacency list, Graph Modeling Language (GML), and Graph Markup Language (GraphML), providing a more extensible framework for manual graph prompts. GraphText [175] translates graph features and relationships into natural language and assembles them to form a Graph-syntax Tree. This Graph-syntax Tree serves as a natural language graph prompt, enabling LLMs to comprehend graph data and engage in interpretable and interactive graph reasoning. In the field of molecules, LLM4Mol [91] uses SMILES (Simplified Molecular Input Line Entry System) to directly describe the property of molecular, which is a widely adopted molecular describing language. Furthermore, InstructGLM [157] employs a technique called instruction prompting. This approach involves the design of a set of graph descriptions centered around a central node, coupled with task-specific descriptions. Graph-LLM [9] also uses natural language instructions and subsequently conducts a series of comprehensive experiments.

5.3.2 Automatic Prompting. Creating prompts manually is a time-consuming task, and these prompts can sometimes be sub-optimal [52]. To address these drawbacks, automatically generated prompts have been introduced for further adaptation. GPT4Graph [29] firstly tries to employ three different types of prompts generated by LLM itself, namely graph summarization, graph exploration and graph completion, in graph tasks. Specifically, graph summarization generates a summary of the given graph by extracting key features or a summary of the neighborhood information of target nodes. Graph exploration means generating a sequence of queries or actions to retrieve information from the given graph. Graph completion generates partial graphs and prompt itself to complete the missing parts. By leveraging these self-prompting strategies, LLMs can actively engage in the understanding and manipulation of graphs, facilitating graph-based reasoning and learning. Graph-LLM [9] uses automatic prompts as well in the form of neighbor summary, and their experimental results once again emphasize the efficiency of automatic prompting.

Additionally, there are various adaptation approaches based on fine-tuning, including Vanilla Fine-Tuning [13], Intermediate Fine-Tuning (IFT) [90], Multi-task Fine-Tuning (MTFT) [75], and Parameter Efficient Fine-Tuning [39]. These methods offer efficient ways to adapt pre-trained models to downstream tasks, although they have not been applied to graph tasks at this time. However, we anticipate that future research will explore the integration of these adaptation approaches into graph tasks, further advancing the development of the graph foundation model.

5.4 Discussion

Efforts of aligning graph data with natural language and using sole LLMs as graph learners has paved the way for exciting developments. The integration of graph data, text, and other modalities into transformer-based models presents a promising way, with the potential to connect techniques from the GNN field with advancements in the LLM domain. Moreover, leveraging LLMs allows for the unification of various graph tasks, as these tasks can all be described in natural language. This makes LLM-based backbones a more competitive selection for building graph foundation models.

Nonetheless, it is essential to acknowledge that the current ways of utilizing LLMs as backbones for graph learning also presents inherent limitations. These limitations encompass the inability of LLMs to perform precise mathematical calculations pertaining to graph properties, their incapacity to engage in multi-hop logical reasoning through graph links, the challenge they face in capturing the topological structures prevalent in highly connected graphs, and their struggle in handling the dynamic nature of graphs that evolve over time [165]. These shortcomings underscore the need for further research and innovation in using LLM-based models for graph learning.

6 GNN+LLM-BASED MODELS

GNN-based models lack the ability to process text and thus cannot directly make predictions based on textual data. Additionally, they cannot make predictions based on natural language instructions provided by users. Consequently, exploring the performance of models with a substantial parameter count in graph-related tasks is imperative. On the other hand, LLM-based models for graph learning have their inherent limitations. These limitations include the incapability of LLMs to process precise mathematical calculations and the inability to handle multi-hop logical reasoning, etc. These shortcomings underline the necessity for further research and innovation in this domain. To overcome these limitations and harness the strengths of both language understanding from LLMs and structural analysis from GNNs, integrating LLMs and GNNs can potentially lead to a more comprehensive and powerful model. We summarize and categorize the works mentioned in this section in Table 4.

6.1 Backbone Architectures

To simultaneously utilize information from both the graph and text and accomplish a variety of tasks, we need to design a framework that effectively integrates LLM and GNN. Depending on the prediction model, GNN+LLM-based methods can be classified as: 1) GNN-centric Methods, 2) Symmetric Methods, and 3) LLM-centric Methods, as illustrated in Figure 5.

6.1.1 GNN-centric Methods. Several works aim to utilize LLM to extract node features from raw data and make predictions using GNN. These approaches are denoted as GNN-centric models. For example, SimTeG [16] performs a parameter-efficient fine-tuning of an LLM on the textual dataset of a TAG (text-attributed graph). The textual dataset T is annotated with task-specific labels $\mathbf{Y}$, where $G = (V, E, T)$ and T is the set of texts with each element aligned with a node in V. Then the downstream task loss for fine-tuning is:

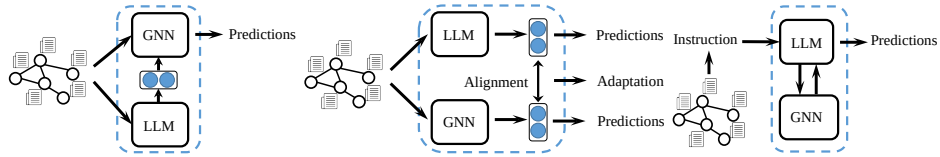
$$\begin{aligned} \text{Loss}_{\text{CLS}} &= \mathcal{L}_\theta(\phi(\text{LLM}(T)), \mathbf{Y}), \\ \text{Loss}_{\text{LINK}} &= \mathcal{L}_\theta\left(\phi\left(\text{LLM}\left(T_{\text{src}}\right), \text{LLM}\left(T_{\text{dst}}\right)\right), \mathbf{Y}\right), \end{aligned} \tag{4}$$

where $\phi(\cdot)$ is the classifier for the classification task or similarity function for the link prediction task, T_{src} and T_{dst} are the texts of the source node and the target node, respectively, Loss_{CLS} and $\text{Loss}_{\text{LINK}}$ are the loss of classification and link prediction task, respectively. Thus we can get the node representations $\mathbf{X}$ with fine-tuned LLM, achieved by removing the head layer. Then we can train GNN with the loss:

$$\begin{aligned} \text{Loss}_{\text{CLS}} &= \mathcal{L}_\theta(\phi(\text{GNN}(\text{LLM}(T))), \mathbf{Y}), \\ \text{Loss}_{\text{LINK}} &= \mathcal{L}_\theta\left(\phi\left(\text{GNN}\left(\text{LLM}(T_{\text{src}})\right), \text{GNN}\left(\text{LLM}(T_{\text{dst}})\right)\right), \mathbf{Y}\right). \end{aligned} \tag{5}$$

For LLMs that do not provide direct access to their embeddings such as ChatGPT, TAPE [35] engages these LLMs through text interactions. Specifically, TAPE first utilizes an LLM to generate a ranked prediction list and explanation based on the original text, and then an LM is utilized and fine-tuned to transform the original text and additional features of predictions and explanation generated by LLM into node features. Subsequently, downstream GNNs can utilize the features for prediction tasks. TAPE extracts graph-agnostic features and cannot capture correlations between graph topology and raw features. To this end, GIANT [11] utilizes a graph-structure aware self-supervised learning method to finetune the LM. Consequently, the text representations encompass graph-related information. Concerning the scalability problem of combining LLM and GNN, GraD [79] concurrently optimizes a GNN teacher and a graph-free student over the nodes of the graph using a shared LLM. This encourages the graph-free student to make use of graph information provided by the GNN teacher while allowing the GNN teacher to effectively utilize textual information from unlabeled nodes. In GALM [147], the focus is on exploring pre-training approaches for models that combine text and graph data, particularly on extensive heterogeneous graphs enriched with rich textual data.

6.1.2 Symmetric Methods. Also, there are some works that align the embeddings of GNN and LLM to make better predictions or utilize the embeddings for other downstream tasks, denoted as symmetric methods. Most GNN-centric based methods involve two sequential steps: text encoding and graph aggregation. It is important to note that during the generation of text embeddings, there is no exchange of information between nodes. To consider the interrelated nature of connected nodes, several works try to utilize GNN and LLM together to get structure-aware text features. GraphFormer [153] fuses text embedding and graph aggregation as an iterative workflow. During each iteration, the interconnected nodes will engage in information exchange within the layerwise GNN component, formulated as $\hat{z}^l = \text{GNN}(z^l)$, where $\mathbf{z}^l$ is the output of l-th layer of GNN.



(a) GNN-centric methods. (b) Symmetric methods. (c) LLM-centric methods.

Fig. 5. An illustration of GNN+LLM-based models. In (b), the aligned embeddings can be further utilized for downstream tasks. In (c), the methods take an instruction as input and output an answer.

As a result, each node will incorporate information from its neighboring nodes. The Transformer component then operates on these enhanced node features, enabling the generation of progressively more informative node representations as $z^{l+1} = \text{TRM}(\text{CONCAT}(\hat{z}^l, h^l))$, where TRM is the transformer, and h^l is the output of l-th layer of transformer. However, this method suffers from scalability issues because the memory complexity is proportional to the graph size as neighborhood texts are also encoded. GLEM [174] employs a variational EM framework to alternatively update the LLMs and GNNs, thus can essentially capture node label distribution conditioned on the local text attributes. In contrast, GNN uses the text and label information of neighboring nodes to predict labels, thus characterizing global conditional label distribution. By doing so, GLEM efficiently incorporates local textual data and global structural information into its components and can ease the scalability issue.

Other studies employ distinct encoders for graph nodes and texts, training them to align their representations within a shared latent space. ConGrat [4] utilizes a contrastive pre-training framework for jointly learning embeddings of graph nodes and texts. G2P2 [136] jointly pre-trains a graph-text model utilizing three graph interaction-based contrastive strategies, and then explores prompting for the downstream tasks. SAFER [6] trains a text encoder and GNN separately to get the embeddings, and concatenates the embeddings to get the final social context-aware embedding for a logistic regression (LR) classifier. Some other works [18, 73, 103, 109] also utilize GNN and LLM to learn representations for molecules. These models employ a contrastive learning strategy to effectively pre-train on a dataset containing pairs of molecular graphs and corresponding textual descriptions. By simultaneously learning the chemical structures of molecules and their associated text through this approach, these models can then be applied to various downstream tasks.

6.1.3 LLM-centric Methods. While LLMs have shown impressive performance in various natural language tasks, they struggle with precise mathematical calculations, multi-step logic reasoning, spatial and topological perception, and handling temporal progression [165]. Hence some works utilize GNNs to enhance the performance of LLM, denoted as LLM-centric methods. For example, Graph-ToolFormer [165] enhances LLMs by incorporating prompts augmented by ChatGPT, seamlessly integrating API calls from external graph learning tools into graph reasoning statements. This enhancement empowers LLMs to handle graph reasoning tasks more effectively, including fundamental graph property reasoning as well as advanced tasks like reasoning about bibliographic paper topics, molecular graph functions and sequential recommender systems.

6.2 Pre-training

To train the model and enable it to handle both graph and text information, we need to train the model on a large amount of data. LLM and GNN can be pre-trained on textual data and graph data respectively. Furthermore, the GNN+LLM-based methods can further be pre-trained on data composed of graph and text. In this subsection, we category the pre-training strategies as GNN or LLM-based, and alignment-based.

Table 4. Details of approaches involved as GNN+LLM-based models.

Model	Backbone Architecture	Pre-training	Adaptation
SimTeG [16]	GNN-centric	MLM, TTCL	Parameter-Efficient FT
TAPE [35]	GNN-centric	LM	Tuning-free Prompting + Parameter-Efficient FT
GIANT [11]	GNN-centric	MLM	Vanilla FT
GraD [79]	GNN-centric	MLM	Parameter-Efficient FT
GALM [147]	GNN-centric	Graph Reconstruction	Vanilla FT
GraphFormer [153]	Symmetric	MLM	Vanilla FT
GLEM [174]	Symmetric	MLM	Vanilla FT
ConGrat [4]	Symmetric	MLM + GTCL	Parameter-Efficient FT
G2P2 [136]	Symmetric	GTCL	Prompt Tuning
SAFER [6]	Symmetric	MLM	Parameter-Efficient FT
Text2Mol [18]	Symmetric	MLM + GTCL	Parameter-Efficient FT
MoMu [109]	Symmetric	MLM + GTCL	Parameter-Efficient FT
MoleculeSTM [73]	Symmetric	MLM + GTCL	Parameter-Efficient FT
CLAMP [103]	Symmetric	MLM + GTCL	Parameter-Efficient FT
Graph-Toolformer [165]	LLM-centric	LM	Tuning-free Prompting + Vanilla FT

6.2.1 GNN or LLM-based. Other synergized LLM and GNN frameworks leverage pre-trained LLMs to obtain text embeddings. The majority of existing models, such as SimTeG [16], GIANT [11], GraD [79], GraphFormer [153], GLEM [174], ConGrat [4], SAFER [6], Text2Mol [18], MoMu [109], MoleculeSTM [73], and CLAMP [103] employ Masked Language Modeling (MLM) during pre-training. Some models, like TAPE and Graph-ToolFormer, opt for Language Modeling (LM) in the pre-training phase. Additionally, SimTeG integrates Text-Text Contrastive Learning (TTCL), a technique that leverages certain observed text pairs exhibiting more semantic similarity than randomly selected pairs during the pre-training phase as:

$$\text{Loss}_{\text{TTCL}} = \mathbf{E}_{x,y^+,y^-}\left[-\log \frac{\exp\left(k\left(x,y^+\right)\right)}{\exp\left(k\left(x,y^+\right)\right)+\exp\left(k\left(x,y^-\right)\right)}\right], \tag{6}$$

where $\mathbf{E}$ is the expectation, k is the score function, y^+ is the positive sample and y^- is the negative sample. Additionally, GALM [147] utilizes graph reconstruction for pre-training on extensive graph datasets, and thus can incorporate the graph information into the pre-trained LLMs.

6.2.2 Alignment-based. Symmetric methods of LLM and GNN like ConGrat [4], Text2Mol [18], MoMu [109], MoleculeSTM [73], and CLAMP [103] are pre-trained on large datasets with Graph-Text Contrastive Learning (GTCL), which aligns the embeddings of the graph encoder and the text encoder. The embeddings involve rich information about graph structure and text, thus demonstrating appealing performance on downstream datasets. For a molecule example, CLAMP minimizes the contrastive loss as below:

$$\text{Loss}_{\text{NCE}} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(k(\text{LLM}(\mu_i), \text{GNN}(\xi_i))) + (1-y_i)\log(1-k(\text{LLM}(\mu_i), \text{GNN}(\xi_i))), \tag{7}$$

where μ_i is the text representation, ξ_i is the graph representation, and k is a score function to predict the activity of a molecule. The contrastive loss promotes the active molecules on a bioassay to have similar embeddings to the embedding of a specific bioassay, while ensuring that inactive molecules have dissimilar embeddings to it.

6.3 Adaptation

The adaptation phase plays a pivotal role in optimizing GNN+LLM-based models for efficient graph learning. Apart from some works [18, 73, 103, 109] which test the model's performance on zero-shot tasks such as zero-shot structure-text retrieval and zero-shot text-based molecule editing, models in most cases need adaptation. In this subsection, we categorize these adaptation strategies into two main types: fine-tuning and prompt-tuning.

6.3.1 Fine-tuning. To adapt to the downstream tasks, some works [11, 153, 174] utilize vanilla fine-tuning methods for node classification tasks. However, vanilla fine-tuning methods involve adjusting a broad range of model parameters, which can be computationally intensive and resource-demanding. So other works [4, 6, 16, 18, 35, 73, 79, 103, 109, 147] utilize parameter-efficient fine-tuning methods for downstream tasks, resulting in a more efficient and resource-friendly approach.

6.3.2 Prompt-Tuning. The prompt-tuning approach is employed in certain studies [35, 136, 165]. In TAPE [35], the initial text features are incorporated into a specialized prompt to interrogate a language model, generating a ranked list of predictions along with explanations. Subsequently, the expanded text features are utilized for finetuning on an LLM. For Graph-ToolFormer [165], a substantial dataset of prompts linked to graph reasoning API call statements is annotated and expanded using an LLM, integrating the most suitable calls to external APIs. This dataset is subsequently employed to fine-tune open-source language models like LLaMA, imparting them with the capability to optimally utilize external graph reasoning tools. G2P2 [136] leverages prompt-tuning to automatically optimize prompts with limited labeled data for efficient adaptation to downstream tasks.

6.4 Discussion

To summarize, LLMs excel in capturing complex linguistic patterns and semantics from textual data, allowing the GNN+LLM-based models to generate embeddings that involve rich text, structure information, and even external knowledge of LLMs, thus leading to better model performance. Also, when integrated with GNN, LLM's reasoning capabilities over graphs may be enhanced. At the same time, these models can also be regarded as multimodal models to accomplish cross-modal tasks, such as text-graph retrieval tasks. The embeddings can be then utilized for a bunch of downstream tasks.

However, training a GNN+LLM-based model often requires large amounts of data (especially paired data) and significant computational resources due to the complexity and scale of both LLMs and GNNs. Effectively integrating LLMs and GNNs in a coherent and effective manner is challenging and requires careful design and experimentation to achieve optimal performance.

7 CHALLENGES AND FUTURE DIRECTIONS

Although the previous sections have discussed the concepts and a lot of related works towards graph foundation models, there are still many avenues for future exploration in this research area. This section will delve into these issues.

7.1 Challenges about Data and Evaluation

7.1.1 Data Quantity and Quality. The improvements in data quantity and data quality are the key factors contributing to the effectiveness of foundation models [2]. At present, there is still a limited amount of open-source large-scale graph data [41, 54], and each dataset is primarily concentrated in a single domain. This poses a challenge to learn graph foundation models for diverse data domains. Hence, it is necessary to collect and organize a unified, massive dataset that covers graph data and related corpora across different domains. Additionally, if the graph data are noisy, incomplete, or not properly curated, it will negatively affect the performance of graph foundation models. To enhance the

data quality of graph foundation models, efforts have been made to propose augmentation strategies from various perspectives, including graph structure learning, feature competion and label mixing, etc. However, since existing data augmentation techniques are typically tailored for individual GNN-based models, there is a need for further exploration on how to effectively augment graph data for LLM-based or GNN+LLM-based models.

7.1.2 Evaluation. With the help of natural language instructions and powerful generation capabilities, LLMs can support a variety of open-ended tasks [118]. This presents new opportunities for graph foundation models based on LLM. However, due to the lack of labels in open-ended tasks, evaluating the performance of GFMs in such tasks is a challenge. When using LLM as a language foundation model, the evaluation of its performance on open-ended tasks has evolved from human evaluation [84] to meta-evaluation [162]. The question of whether existing LLM evaluation methods [84, 162] can be applied to GFMs remains to be explored. Beyond evaluating the performance of graph foundation models, it is also worth evaluating their robustness, trustworthiness, or holistic performance, similar to the current practices for language foundation models [3, 123, 124].

7.2 Challenges about Model

7.2.1 Model Architectures. As mentioned above, the designs of backbone architectures and learning paradigms are crucial for the implementation of graph foundation models. Although this article has outlined some potential solutions, it does not rule out the possibility of better ones. For example, regarding the backbone architecture, recent works have proposed model architectures that go beyond the Transformer [114, 161], offering improved performance [114] or interpretability [161]. However, it is still unknown whether these backbone architectures can be used for dealing with graph data. Additionally, when utilizing GNN+LLM-based models, it is worth exploring how to more effectively align the outputs of both models. Furthermore, there is limited research regarding the emergent abilities of GNN-based [46] or LLM-based [126] graph foundation models. It is yet unclear whether GNN+LLM-based models may have greater potential for emergence. Given that current multimodal foundation models [22] primarily handle text, images, audio, and other modalities, it is an interesting research direction to explore whether GNNs can be employed to further expand the diversity of modalities covered by multimodal foundation models.

7.2.2 Model Training. In order to achieve homogeneity and make effective use of pre-training data, it is crucial to design appropriate pretext tasks in pre-training. Unlike many language foundation models, which often use LM [84] or MLM [13] as pretext tasks, there are now various forms of pretext tasks tailored to different GFM model architectures. Whether each type of pretext task has its own applicable scope and whether there will be a unified pretext task is worth further exploration. After obtaining pre-trained graph foundation models, adapting them to downstream tasks is a vital concern. Apart from fine-tuning and prompting that are introduced in this article, there are other potential techniques that can be applied, such as knowledge distillation [51], reinforcement learning from human feedback (RLHF) [84] and model editing [156]. Whether the above-mentioned techniques can be applied to graph foundation models will be a focal point of future research.

7.3 Challenges about Applications

7.3.1 Killer Applications. In comparison to the outstanding performance of language foundation models in tasks like text translation [36] and text generation [53], whether graph foundation models can similarly catalyze groundbreaking applications in graph tasks is not yet clear. For scenarios that are well-suited for the application of GNNs, such as e-commerce [163] and finance [127], there is hope that training graph foundation models combined with LLMs can better support open-ended tasks [118]. Furthermore, graph foundation models have the potential to make breakthroughs in

some emerging fields. For example, drug development is a time-consuming and costly process [139], and language foundation models have already been successfully used for related tasks like target identification and side effect prediction [2]. Given the 3D geometric structure of proteins [69, 70, 74], graph foundation models hold the promise of enhancing the drug discovery pipeline by leveraging their ability to model graph structure information [145], potentially speeding up the process further. Additionally, urban computing may also represent a crucial application scenario for graph foundation models. It is worth noting that traditional traffic prediction techniques have been primarily focused on addressing individual tasks such as travel demand prediction [182] and traffic flow prediction [131], lacking a comprehensive understanding of the entire transportation system. Given that the transportation system can be viewed as a spatio-temporal graph, graph foundation models hold the potential to capture the participation behavior of actors in the transportation system [132], thereby offering a unified approach to addressing various issues in urban computing.

7.3.2 Safety. Despite the strong performance of LLM-based foundation models, their black-box nature [112] introduces a host of safety concerns, such as hallucination and privacy leaks. The hallucination refers to the output appearing plausible but deviating from user input, context, or facts [43, 171]. Existing research suggests that this phenomenon is associated with multiple factors, such as the model's overconfidence in its own behavior [98] and the misunderstanding of false correlations [67]. However, it is currently uncertain whether graph foundation models might also suffer from hallucination. Given the unique nature of graph data, we may require certain techniques to prevent or mitigate hallucination, such as confidence calibration [130] or counterfactual reasoning [115]. Additionally, given that existing research has indicated privacy risks in both GNN [141, 169] and LLM [107], enhancing the privacy of graph foundation models is also a critical concern. Some potential solutions include federated learning [8, 140], RLHF [84] and red teaming [25], but whether these methods can be applied to graph foundation models is still unknown.

8 CONCLUSIONS

The development of foundation models and graph machine learning has spurred the emergence of a new research direction, with the aim to train on broad graph data and apply it to a wide range of downstream graph tasks. In this article, we propose the concept of graph foundation models (GFMs) for the first time, and provide an introduction to relevant concepts and representative methods. We summarize existing works towards GFMs into three main categories based on their reliance on graph neural networks (GNNs) and large language models (LLMs): GNN-based models, LLM-based models, and GNN+LLM-based models. For each category of methods, we introduce their backbone architectures, pre-training, and adaptation strategies separately. After providing a comprehensive overview of the current landscape of graph foundation models, this article also points out the future directions for this evolving field.

REFERENCES

[1] Uri Alon and Eran Yahav. 2020. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *Proc. of ICLR*.

[2] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* (2021).

[3] Rishi Bommasani, Percy Liang, and Tony Lee. 2023. Holistic Evaluation of Language Models. *Annals of the New York Academy of Sciences* (2023).

[4] William Brannon, Suyash Fulay, Hang Jiang, Wonjune Kang, Brandon Roy, Jad Kabbara, and Deb Roy. 2023. ConGraT: Self-Supervised Contrastive Pretraining for Joint Graph and Text Embeddings. *arXiv preprint arXiv:2305.14321* (2023).

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Proc. of NeurIPS* (2020).
[6] Shantanu Chandra, Pushkar Mishra, Helen Yannakoudakis, Madhav Nimishakavi, Marzieh Saeidi, and Ekaterina Shutova. 2020. Graph-based modeling of online communities for fake news detection. *arXiv preprint arXiv:2008.06274* (2020).
[7] Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. 2022. Structure-aware transformer for graph representation learning. In *Proc. of ICML*.
[8] Huiqiang Chen, Tianqing Zhu, Tao Zhang, Wanlei Zhou, and Philip S Yu. 2023. Privacy and Fairness in Federated Learning: on the Perspective of Trade-off. *Comput. Surveys* (2023).
[9] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. 2023. Exploring the potential of large language models (llms) in learning on graphs. *arXiv preprint arXiv:2307.03393* (2023).
[10] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)* (2023).
[11] Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S. Dhillon. 2022. Node Feature Extraction by Self-Supervised Multi-scale Neighborhood Prediction. In *Proc. of ICLR*.
[12] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311* (2022).
[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2019).
[14] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234* (2022).
[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. (2021).
[16] Keyu Duan, Qian Liu, Tat-Seng Chua, Shuicheng Yan, Wei Tsang Ooi, Qizhe Xie, and Junxian He. 2023. Simteg: A frustratingly simple approach improves textual graph learning. *arXiv preprint arXiv:2308.02565* (2023).
[17] Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. 2022. Long range graph benchmark. *Proc. of NeurIPS* (2022).
[18] Carl Edwards, ChengXiang Zhai, and Heng Ji. 2021. Text2mol: Cross-modal molecule retrieval with natural language queries. In *Proc. of EMNLP*.
[19] Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. 2010. Why does unsupervised pre-training help deep learning?. In *Proc. of AISTATS*.
[20] Wenqi Fan, Zihuai Zhao, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Jiliang Tang, and Qing Li. 2023. Recommender systems in the era of large language models (llms). *arXiv preprint arXiv:2307.02046* (2023).
[21] Taoran Fang, Yunchao Zhang, Yang Yang, Chunping Wang, and Lei Chen. 2022. Universal Prompt Tuning for Graph Neural Networks. *arXiv preprint arXiv:2209.15240* (2022).
[22] Nanyi Fei, Zhiwu Lu, Yizhao Gao, Guoxing Yang, Yuqi Huo, Jingyuan Wen, Haoyu Lu, Ruihua Song, Xin Gao, Tao Xiang, et al. 2022. Towards artificial general intelligence via a multimodal foundation model. *Nature Communications* (2022).
[23] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proc. of AAAI*.
[24] Linton Freeman. 2004. The development of social network analysis. *A Study in the Sociology of Science* (2004).
[25] Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. 2022. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858* (2022).
[26] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *Proc. of ICML*.
[27] Xumeng Gong, Cheng Yang, and Chuan Shi. 2023. Ma-gcl: Model augmentation tricks for graph contrastive learning. In *Proc. of AAAI*.
[28] Anchun Gui, Jinqiang Ye, and Han Xiao. 2023. G-Adapter: Towards Structure-Aware Parameter-Efficient Transfer Learning for Graph Transformer Networks. *arXiv preprint arXiv:2305.10329* (2023).
[29] Jiayan Guo, Lun Du, and Hengyu Liu. 2023. GPT4Graph: Can Large Language Models Understand Graph Structured Data? An Empirical Evaluation and Benchmarking. *arXiv preprint arXiv:2305.15066* (2023).

[30] Yuxin Guo, Cheng Yang, Yuluo Chen, Jixi Liu, Chuan Shi, and Junping Du. 2023. A Data-centric Framework to Endow Graph Neural Networks with Out-Of-Distribution Detection Ability. In *Proc. of KDD*.
[31] Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. Warp: Word-level adversarial reprogramming. *arXiv preprint arXiv:2101.00121* (2021).
[32] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Proc. of NeurIPS* (2017).
[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proc. of CVPR*.
[34] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. DEBERTA: DECODING-ENHANCED BERT WITH DISENTANGLED ATTENTION. In *Proc. of ICLR*.
[35] Xiaoxin He, Xavier Bresson, Thomas Laurent, and Bryan Hooi. 2023. Explanations as Features: LLM-Based Features for Text-Attributed Graphs. *arXiv preprint arXiv:2305.19523* (2023).
[36] Amr Hendy, Mohamed Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, and Hany Hassan Awadalla. 2023. How good are gpt models at machine translation? a comprehensive evaluation. *arXiv preprint arXiv:2302.09210* (2023).
[37] Zhenyu Hou, Yufei He, Yukuo Cen, Xiao Liu, Yuxiao Dong, Evgeny Kharlamov, and Jie Tang. 2023. GraphMAE2: A Decoding-Enhanced Masked Self-Supervised Graph Learner. In *Proceedings of the ACM Web Conference 2023*.
[38] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. Graphmae: Self-supervised masked graph autoencoders. In *Proc. of KDD*.
[39] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proc. of ICML*.
[40] Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proc. of ACL*.
[41] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Proc. of NeurIPS* (2020).
[42] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. In *Proc. of ICLR*.
[43] Xuming Hu, Junzhe Chen, Xiaochuan Li, Yufei Guo, Lijie Wen, Philip S. Yu, and Zhijiang Guo. 2023. Do Large Language Models Know about Facts? arXiv:cs.CL/2310.05177
[44] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. Gpt-gnn: Generative pre-training of graph neural networks. In *Proc. of KDD*.
[45] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proc. of WWW*.
[46] Qian Huang, Hongyu Ren, Peng Chen, Gregor Kržmanc, Daniel Zeng, Percy Liang, and Jure Leskovec. 2023. PRODIGY: Enabling In-context Learning Over Graphs. *arXiv preprint arXiv:2305.12600* (2023).
[47] Weiwei Jiang and Jiayun Luo. 2022. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications* (2022).
[48] Xiaodong Jiang, Pengsheng Ji, and Sheng Li. 2019. Censnet: convolution with edge-node switching in graph neural networks. In *Proc. of IJCAI*.
[49] Xunqiang Jiang, Tianrui Jia, Yuan Fang, Chuan Shi, Zhe Lin, and Hui Wang. 2021. Pre-training on large-scale heterogeneous graph. In *Proc. of KDD*.
[50] Xunqiang Jiang, Yuanfu Lu, Yuan Fang, and Chuan Shi. 2021. Contrastive pre-training of GNNs on heterogeneous graphs. In *Proc. of CIKM*.
[51] Yuxin Jiang, Chunkit Chan, Mingyang Chen, and Wei Wang. 2023. Lion: Adversarial Distillation of Closed-Source Large Language Model. *arXiv preprint arXiv:2305.12870* (2023).
[52] Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics* (2020).
[53] Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858* (2019).
[54] Arpandeep Khatua, Vikram Sharma Mailthody, Bhagyashree Taleka, Tengfei Ma, Xiang Song, and Wen-mei Hwu. 2023. IGB: Addressing The Gaps In Labeling, Features, Heterogeneity, and Size of Public Graph Datasets for Deep Learning Research. *arXiv preprint arXiv:2302.13522* (2023).
[55] Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. 2022. Pure transformers are powerful graph learners. In *Proc. of NeurIPS*.
[56] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).
[57] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proc. of ICLR*.

[58] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. 2021. Rethinking graph transformers with spectral attention. In *Proc. of NeurIPS*.
[59] John Boaz Lee, Ryan Rossi, and Xiangnan Kong. 2018. Graph classification using structural attention. In *Proc. of KDD*.
[60] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proc. of ACL*.
[61] Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, et al. 2021. Datasets: A Community Library for Natural Language Processing. In *Proc. of EMNLP*.
[62] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. 2019. Deepgcns: Can gcns go as deep as cnns?. In *Proc. of ICCV*.
[63] Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. 2023. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. In *Proc. of ICML*.
[64] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. 2020. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Proc. of NeurIPS* (2020).
[65] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proc. of AAAI*.
[66] Shengrui Li, Xueting Han, and Jing Bai. 2023. AdapterGNN: Efficient Delta Tuning Improves Generalization Ability in Graph Neural Networks. *arXiv preprint arXiv:2304.09595* (2023).
[67] Shaobo Li, Xiaoguang Li, Lifeng Shang, Zhenhua Dong, Chengjie Sun, Bingquan Liu, Zhenzhou Ji, Xin Jiang, and Qun Liu. 2022. How pre-trained language models capture factual knowledge? a causal-inspired analysis. *arXiv preprint arXiv:2203.16747* (2022).
[68] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proc. of ACL*.
[69] Xuan Lin, Lichang Dai, Yafang Zhou, Zu-Guo Yu, Wen Zhang, Jian-Yu Shi, Dong-Sheng Cao, Li Zeng, Haowen Chen, Bosheng Song, et al. 2023. Comprehensive evaluation of deep and graph learning on drug–drug interactions prediction. *Briefings in Bioinformatics* (2023).
[70] Xuan Lin, Zhe Quan, Zhi-Jie Wang, Yan Guo, Xiangxiang Zeng, and S Yu Philip. 2022. Effectively Identifying Compound-Protein Interaction Using Graph Neural Representation. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2022).
[71] Chang Liu and Bo Wu. 2023. Evaluating Large Language Models on Graphs: Performance Insights and Comparative Analysis. *arXiv preprint arXiv:2308.11224* (2023).
[72] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *Comput. Surveys* (2023).
[73] Shengchao Liu, Weili Nie, Chengpeng Wang, Jiarui Lu, Zhuoran Qiao, Ling Liu, Jian Tang, Chaowei Xiao, and Anima Anandkumar. 2022. Multi-modal molecule structure-text model for text-based retrieval and editing. *arXiv preprint arXiv:2212.10789* (2022).
[74] Shengchao Liu, Hanchen Wang, Weiyang Liu, Joan Lasenby, Hongyu Guo, and Jian Tang. 2021. Pre-training Molecular Graph Representation with 3D Geometry. In *Proc. of ICLR*.
[75] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504* (2019).
[76] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and Philip S. Yu. 2022. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering* (2022).
[77] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proc. of ICCV*.
[78] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. 2023. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM Web Conference 2023*.
[79] Costas Mavromatis, Vassilis N. Ioannidis, Shen Wang, Da Zheng, Soji Adeshina, Jun Ma, Han Zhao, Christos Faloutsos, and George Karypis. 2023. Train Your Own GNN Teacher: Graph-Aware Distillation on Textual Graphs. In *Proc. of KDD*.
[80] Erxue Min, Runfa Chen, Yatao Bian, Tingyang Xu, Kangfei Zhao, Wenbing Huang, Peilin Zhao, Junzhou Huang, Sophia Ananiadou, and Yu Rong. 2022. Transformer for graphs: An overview from architecture perspective. *arXiv preprint arXiv:2202.08455* (2022).
[81] Luis Müller, Mikhail Galkin, Christopher Morris, and Ladislav Rampásek. 2023. Attending to Graph Transformers. *CoRR* (2023).
[82] Giulia Muzio, Leslie O'Bray, and Karsten Borgwardt. 2021. Biological network analysis with deep learning. *Briefings in bioinformatics* (2021).

[83] OpenAI. 2023. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774* (2023).
[84] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Proc. of NeurIPS* (2022).
[85] Gerhard Paaß and Sven Giesselbach. 2023. *Foundation Models for Natural Language Processing: Pre-trained Language Models Integrating Media*. Springer Nature.
[86] Jeff Z Pan, Simon Razniewski, Jan-Christoph Kalo, Sneha Singhania, Jiaoyan Chen, Stefan Dietze, Hajira Jabeen, Janna Omeliyanenko, Wen Zhang, Matteo Lissandrini, et al. 2023. Large Language Models and Knowledge Graphs: Opportunities and Challenges. *arXiv preprint arXiv:2308.06374* (2023).
[87] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2023. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *arXiv preprint arXiv:2306.08302* (2023).
[88] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proc. of KDD*.
[89] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066* (2019).
[90] Clifton Poth, Jonas Pfeiffer, Andreas Rücklé, and Iryna Gurevych. 2021. What to Pre-Train on? Efficient Intermediate Task Selection. In *Proc. of EMNLP*.
[91] Chen Qian, Huayi Tang, Zhirui Yang, Hong Liang, and Yong Liu. 2023. Can Large Language Models Empower Molecular Property Prediction? *arXiv preprint arXiv:2307.07443* (2023).
[92] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proc. of KDD*.
[93] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences* (2020).
[94] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* (2020).
[95] Md Khaledur Rahman and Ariful Azad. 2021. A Comprehensive Analytical Survey on Unsupervised and Semi-Supervised Graph Representation Learning Methods. *arXiv preprint arXiv:2112.10372* (2021).
[96] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. 2022. Recipe for a general, powerful, scalable graph transformer. *Proc. of NeurIPS* (2022).
[97] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proc. of EMNLP*.
[98] Ruiyang Ren, Yuhao Wang, Yingqi Qu, Wayne Xin Zhao, Jing Liu, Hao Tian, Hua Wu, Ji-Rong Wen, and Haifeng Wang. 2023. Investigating the factual knowledge boundary of large language models with retrieval augmentation. *arXiv preprint arXiv:2307.11019* (2023).
[99] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. 2020. Self-supervised graph transformer on large-scale molecular data. *Proc. of NeurIPS* (2020).
[100] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2020. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *Proc. of ICLR*.
[101] Nikunj Saunshi, Orestis Plevrakis, Sanjeev Arora, Mikhail Khodak, and Hrishikesh Khandeparkar. 2019. A Theoretical Analysis of Contrastive Unsupervised Representation Learning. In *Proc. of ICML*.
[102] Satu Elisa Schaeffer. 2007. Graph clustering. *Computer science review* (2007).
[103] Philipp Seidl, Andreu Vall, Sepp Hochreiter, and Günter Klambauer. 2023. Enhancing Activity Prediction Models in Drug Discovery with the Ability to Understand Human Language. In *Proc. of ICML*.
[104] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and Philip S. Yu. 2016. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering* (2016).
[105] Joakim Skarding, Bogdan Gabrys, and Katarzyna Musial. 2021. Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. *IEEE Access* (2021).
[106] Zixing Song, Xiangli Yang, Zenglin Xu, and Irwin King. 2022. Graph-based semi-supervised learning: A comprehensive review. *IEEE Transactions on Neural Networks and Learning Systems* (2022).
[107] Robin Staab, Mark Vero, Mislav Balunović, and Martin Vechev. 2023. Beyond Memorization: Violating Privacy Via Inference with Large Language Models. *arXiv preprint arXiv:2310.07298* (2023).
[108] Bing Su, Dazhao Du, Zhao Yang, Yujie Zhou, Jiangmeng Li, Anyi Rao, Hao Sun, Zhiwu Lu, and Ji-Rong Wen. 2022. A molecular multimodal foundation model associating molecule graphs with natural language. *arXiv preprint arXiv:2209.05481* (2022).
[109] Bing Su, Dazhao Du, Zhao Yang, Yujie Zhou, Jiangmeng Li, Anyi Rao, Hao Sun, Zhiwu Lu, and Ji-Rong Wen. 2022. A molecular multimodal foundation model associating molecule graphs with natural language. *arXiv preprint*

arXiv:2209.05481 (2022).
[110] Mengying Sun, Jing Xing, Huijun Wang, Bin Chen, and Jiayu Zhou. 2021. MoCL: data-driven molecular fingerprint via knowledge-aware contrastive learning from molecular graph. In *Proc. of KDD*.
[111] Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. 2022. Gppt: Graph pre-training and prompt tuning to generalize graph neural networks. In *Proc. of KDD*.
[112] Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022. Black-box tuning for language-model-as-a-service. In *Proc. of ICML*.
[113] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. 2023. All in One: Multi-Task Prompting for Graph Neural Networks. In *Proc. of KDD*.
[114] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. 2023. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621* (2023).
[115] Juntao Tan, Shijie Geng, Zuohui Fu, Yingqiang Ge, Shuyuan Xu, Yunqi Li, and Yongfeng Zhang. 2022. Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning. In *Proceedings of the ACM Web Conference 2022*.
[116] Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A Large Language Model for Science. *CoRR* (2022).
[117] Josephine Maria Thomas, Alice Moallemy-Oureh, Silvia Beddar-Wiesing, and Clara Holzhüter. 2023. Graph Neural Networks Designed for Different Graph Types: A Survey. *Trans. Mach. Learn. Res.* (2023).
[118] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *CoRR* (2023).
[119] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Proc. of NeurIPS* (2017).
[120] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *Proc. of ICLR*.
[121] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep graph infomax. *arXiv preprint arXiv:1809.10341* (2018).
[122] Lilapati Waikhom and Ripon Patgiri. 2021. Graph neural networks: Methods, applications, and opportunities. *arXiv preprint arXiv:2108.10733* (2021).
[123] Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, et al. 2023. DecodingTrust: A Comprehensive Assessment of Trustworthiness in GPT Models. *arXiv preprint arXiv:2306.11698* (2023).
[124] Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li. 2021. Adversarial glue: A multi-task benchmark for robustness evaluation of language models. *arXiv preprint arXiv:2111.02840* (2021).
[125] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Attributed graph clustering: a deep attentional embedding approach. In *Proc. of IJCAI*.
[126] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can Language Models Solve Graph Problems in Natural Language? *arXiv preprint arXiv:2305.10037* (2023).
[127] Jianian Wang, Sheng Zhang, Yanghua Xiao, and Rui Song. 2021. A review on graph neural network methods in financial applications. *arXiv preprint arXiv:2111.15367* (2021).
[128] Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, et al. 2023. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *arXiv preprint arXiv:2305.11175* (2023).
[129] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *Proc. of WWW*.
[130] Xiao Wang, Hongrui Liu, Chuan Shi, and Cheng Yang. 2021. Be confident! towards trustworthy graph neural networks via confidence calibration. *Proc. of NeurIPS* (2021).
[131] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. 2020. Traffic flow prediction via spatial temporal graph neural network. In *Proc. of WWW*.
[132] Xuhong Wang, Ding Wang, Liang Chen, and Yilun Lin. 2023. Building Transportation Foundation Model via Generative Graph Transformer. *arXiv preprint arXiv:2305.14826* (2023).
[133] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent Abilities of Large Language Models. *Trans. Mach. Learn. Res.* (2022).
[134] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Proc. of NeurIPS* (2022).

[135] Boris Weisfeiler and Andrei Leman. 1968. The reduction of a graph to canonical form and the algebra which appears therein. *nti, Series* (1968).
[136] Zhihao Wen and Yuan Fang. 2023. Augmenting Low-Resource Text Classification with Graph-Grounded Pre-training and Prompting. In *Proc. of SIGIR*.
[137] Zhihao Wen, Yuan Fang, Yihan Liu, Yang Guo, and Shuji Hao. 2023. Voucher Abuse Detection with Prompt-based Fine-tuning on Graph Neural Networks. In *Proc. of CIKM*.
[138] Oliver Wieder, Stefan Kohlbacher, Mélaine Kuenemann, Arthur Garon, Pierre Ducrot, Thomas Seidel, and Thierry Langer. 2020. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies* (2020).
[139] Olivier J Wouters, Martin McKee, and Jeroen Luyten. 2020. Estimated research and development investment needed to bring a new medicine to market, 2009-2018. *Jama* (2020).
[140] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. 2021. Fedgnn: Federated graph neural network for privacy-preserving recommendation. *arXiv preprint arXiv:2102.04925* (2021).
[141] Fan Wu, Yunhui Long, Ce Zhang, and Bo Li. 2022. Linkteller: Recovering private edges from graph neural networks via influence analysis. In *Proc. of SP*.
[142] Lirong Wu, Haitao Lin, Cheng Tan, Zhangyang Gao, and Stan Z Li. 2021. Self-supervised learning on graphs: Contrastive, generative, or predictive. *IEEE Transactions on Knowledge and Data Engineering* (2021).
[143] Likang Wu, Zhaopeng Qiu, Zhi Zheng, Hengshu Zhu, and Enhong Chen. 2023. Exploring large language model for graph data understanding in online job recommendations. *arXiv preprint arXiv:2307.05722* (2023).
[144] Jun Xia, Yanqiao Zhu, Yuanqi Du, and Stan Z Li. 2022. A survey of pretraining on graphs: Taxonomy, methods, and applications. *arXiv preprint arXiv:2202.07893* (2022).
[145] Jun Xia, Yanqiao Zhu, Yuanqi Du, Yue Liu, and Stan Z Li. 2022. A systematic survey of molecular pre-trained models. *arXiv preprint arXiv:2210.16484* (2022).
[146] Shunxin Xiao, Shiping Wang, Yuanfei Dai, and Wenzhong Guo. 2022. Graph neural networks in node classification: survey and evaluation. *Machine Vision and Applications* (2022).
[147] Han Xie, Da Zheng, Jun Ma, Houyu Zhang, Vassilis N. Ioannidis, Xiang Song, Qing Ping, Sheng Wang, Carl J. Yang, Yi Xu, Belinda Zeng, and Trishul Chilimbi. 2023. Graph-Aware Language Model Pre-Training on a Large Graph Corpus Can Help Multiple Graph Applications. In *Proc. of KDD*.
[148] Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. 2022. Self-supervised learning of graph neural networks: A unified review. *IEEE transactions on pattern analysis and machine intelligence* (2022).
[149] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *Proc. of ICLR*.
[150] Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-Jiang Zhang, Qiang Yang, and Stephen Lin. 2006. Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE transactions on pattern analysis and machine intelligence* (2006).
[151] Cheng Yang, Jiawei Liu, and Chuan Shi. 2021. Extract the knowledge of graph neural networks and go beyond it: An effective knowledge distillation framework. In *Proc. of WWW*.
[152] Hongxia Yang. 2019. Aligraph: A comprehensive graph neural network platform. In *Proc. of KDD*.
[153] Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. 2021. GraphFormers: GNN-nested transformers for representation learning on textual graph. *Proc. of NeurIPS* (2021).
[154] Ling Yang, Jiayi Zheng, Heyuan Wang, Zhongyi Liu, Zhilin Huang, Shenda Hong, Wentao Zhang, and Bin Cui. 2023. Individual and Structural Graph Information Bottlenecks for Out-of-Distribution Generalization. *IEEE Transactions on Knowledge and Data Engineering* (2023).
[155] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Proc. of NeurIPS* (2019).
[156] Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing Large Language Models: Problems, Methods, and Opportunities. *arXiv preprint arXiv:2305.13172* (2023).
[157] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2023. Natural language is all a graph needs. *arXiv preprint arXiv:2308.07134* (2023).
[158] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation? *Proc. of NeurIPS* (2021).
[159] Jiaxuan You, Zhitao Ying, and Jure Leskovec. 2020. Design space for graph neural networks. *Proc. of NeurIPS* (2020).
[160] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Proc. of NeurIPS* (2020).

[161] Yaodong Yu, Sam Buchanan, Druv Pai, Tianzhe Chu, Ziyang Wu, Shengbang Tong, Benjamin D Haeffele, and Yi Ma. 2023. White-Box Transformers via Sparse Rate Reduction. *arXiv preprint arXiv:2306.01129* (2023).
[162] Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. 2023. Evaluating Large Language Models at Evaluating Instruction Following. *arXiv preprint arXiv:2310.07641* (2023).
[163] Dalong Zhang, Xin Huang, Ziqi Liu, Jun Zhou, Zhiyang Hu, Xianzheng Song, Zhibang Ge, Lin Wang, Zhiqiang Zhang, and Yuan Qi. 2020. AGL: A Scalable System for Industrial-purpose Graph Machine Learning. *Proc. VLDB Endow.* (2020).
[164] Hang Zhang, Xin Li, and Lidong Bing. 2023. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858* (2023).
[165] Jiawei Zhang. 2023. Graph-ToolFormer: To Empower LLMs with Graph Reasoning Ability via Prompt Augmented by ChatGPT. *arXiv preprint arXiv:2304.11116* (2023).
[166] Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. 2020. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140* (2020).
[167] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *Proc. of ICML*.
[168] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. In *Proc. of ICONIP*.
[169] Mengchun Zhang, Maryam Qamar, Taegoo Kang, Yuna Jung, Chenshuang Zhang, Sung-Ho Bae, and Chaoning Zhang. 2023. A survey on graph diffusion models: Generative ai in science for molecule, protein and material. *arXiv preprint arXiv:2304.01565* (2023).
[170] Yiyuan Zhang, Kaixiong Gong, Kaipeng Zhang, Hongsheng Li, Yu Qiao, Wanli Ouyang, and Xiangyu Yue. 2023. Meta-transformer: A unified framework for multimodal learning. *arXiv preprint arXiv:2307.10802* (2023).
[171] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023. Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models. *arXiv preprint arXiv:2309.01219* (2023).
[172] Ziwei Zhang, Haoyang Li, Zeyang Zhang, Yijian Qin, Xin Wang, and Wenwu Zhu. 2023. Large Graph Models: A Perspective. *arXiv preprint arXiv:2308.14522* (2023).
[173] Haiteng Zhao, Shengchao Liu, Chang Ma, Hannan Xu, Jie Fu, Zhi-Hong Deng, Lingpeng Kong, and Qi Liu. 2023. GIMLET: A Unified Graph-Text Model for Instruction-Based Molecule Zero-Shot Learning. *bioRxiv* (2023).
[174] Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2023. Learning on Large-scale Text-attributed Graphs via Variational Inference. In *Proc. of ICLR*.
[175] Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. 2023. GraphText: Graph Reasoning in Text Space. arXiv:cs.CL/2310.01089
[176] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).
[177] Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, et al. 2023. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *arXiv preprint arXiv:2302.09419* (2023).
[178] Dawei Zhou, Lecheng Zheng, Dongqi Fu, Jiawei Han, and Jingrui He. 2022. Mentorgnn: Deriving curriculum for pre-training gnns. In *Proc. of CIKM*.
[179] Ronghang Zhu, Xiaodong Jiang, Jiasen Lu, and Sheng Li. 2021. Cross-domain graph convolutions for adversarial unsupervised domain adaptation. *IEEE Transactions on Neural Networks and Learning Systems* (2021).
[180] Yun Zhu, Jianhao Guo, and Siliang Tang. 2023. SGL-PT: A Strong Graph Learner with Graph Prompt Tuning. *arXiv preprint arXiv:2302.12449* (2023).
[181] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131* (2020).
[182] Dingyi Zhuang, Shenhao Wang, Haris Koutsopoulos, and Jinhua Zhao. 2022. Uncertainty quantification of sparse travel demand prediction with spatial-temporal graph neural networks. In *Proc. of KDD*.
[183] Xiandong Zou, Xiangyu Zhao, Pietro Liò, and Yiren Zhao. 2023. Will More Expressive Graph Neural Networks do Better on Generative Tasks? *arXiv preprint arXiv:2308.11978* (2023).