Exploring the potential of language models for graph learning: opportunities and challenges

Yuqun Wang[1,*], Libin Chen[1], Qian Li[1], and Hongfu Liu[1]

[1] *College of Intelligence Science and Technology, National University of Defense Technology, Changsha, China*

Abstract

Graph learning methods are becoming increasingly popular in solving problems related to social networks, biological networks, and other real-world applications. With the rapid development of large language models (LLMs), they are also being used for graph-related tasks and combined with traditional graph neural network (GNN)-based approaches to improve the ability to process graphs associated with text and graph-structured data. In this paper, we provide a review and analyse existing approaches. Firstly, we propose a new taxonomy that classifies existing methods into three categories based on LLMs and GNNs who serve as the final task solving component. Based on this, representative models among them are summarised for each category. Finally, we analyse the limitations of the existing methods and provide an outlook on future research directions in this area.

Keywords

Large Language Models, Graph Neural Networks, Natural Language Processing, Graph Learning

1. Introduction

Graph data, found in diverse forms such as the Internet, traffic networks, social networks, and biological networks, can be effectively represented through graphs. The analysis and mining of this data have become pivotal areas of research. Graph neural networks based on deep learning graph modeling methods have become a new field developed in recent years based on traditional neural networks. These networks are able to better overcome the limitations imposed by traditional deep neural network learning by defining suitable neural network models on graph data and applying the deep learning approach to graph data.

However, in the real world, more and more nodes or connecting edges of graphs are associated with attributes in the form of text. Yet some existing graph neural network methods still have some limitations in dealing with textual attributes of nodes in graph

ICCIC 2024: International Conference on Computer and Intelligent Control, June 29–30, 2024, Kuala Lumpur, Malaysia

* Corresponding author.

wangyuqun18@nudt.edu.cn (Y. Wang); chenlibin@nudt.edu.cn (L. Chen);liqian.nudt@nudt.edu.cn (Q. Li); liuhongfu@nudt.edu.cn (H. Liu)

0009-0008-5787-877 (Y. Wang); 0000-0002-7558-865X (L. Chen); 0009-0002-8920-305 (Q. Li); 0000-0002-1459-8424 (H. Liu)

data. Traditional graph neural network methods mainly model the inputs or outputs consisting of nodes and edges on a graph, but ignore the textual attributes contained in the nodes themselves and are unable to model the original textual information.

The advent of Large Language Models (LLMs) has brought new perspectives to this challenge. Large-scale language models, capable of harnessing vast amounts of data, offer strong language understanding and generalization capabilities. As a deep learning-based natural language processing model, LLMs are trained on extensive corpora, enabling them to handle a variety of linguistic tasks. The launch of GPT-3 in 2020 garnered significant attention from scholars towards LLMs, sparking the question: Can Large Language Models leverage their potential in graph learning to overcome the limitations of traditional graph neural network approaches? Although this question has been studied and explored by scholars, systematic research reviews examining the impact of large language models on graph learning remain sparse.

Liu et al. [1], inspired by the foundational roles of LLMs in natural language processing and GNNs in graph data processing, proposed the concept of 'graph foundation models' and provided a definition. Li et al. [2] investigated the advancements and potential future directions of large language models in graph-related tasks. This article aims to explore and summarize the rapidly evolving field, offering an overview of the influence of language models on graph learning for those interested in pursuing research in this area.

Contributions. The main contributions of this paper are summarised as follows.(1) We present the findings of research in the field through a structured taxonomy, which categorizes the existing studies into three distinct classes. (2) Systematic methodological review. For different classification methods, we summarise representative models, describe each model in more detail, and summarise their strengths and weaknesses as well as limitations. (3) Future directions. We provide an in-depth discussion of the limitations of the current work and suggest possible directions for future development in the field.

2. Preliminarys

In this section, we introduce the definition of a correlation graph and formalise the concepts and definitions related to the two key areas of large language models and graph neural networks and their development.

2.1 Definition

- Definition 1(Graph).

A graph is a collection of nodes and edges. A graph is denoted by $G = (V, E)$, where is the set of nodes and is the set of edges. In an undirected graph, edges can be viewed as unordered pairs connecting two nodes; in a directed graph, edges can be viewed as ordered pairs connecting a start node and an end node.

- Definition 2(Text Attribute Graph (TAG)).

For a textual attribute graph, each node is associated with a contiguous textual feature (sentence). the form of the TAG can be represented as $G = (V, E, D)$, where each $v_i \in V$ is associated with some textual information $d_{v_i} \in D$.

2.2 Graph Neural Networks

Graph Neural Networks are neural network architectures designed to solve tasks related on graph-structured data. The basic idea is to iteratively update the representation of a node by combining the representations of its neighbours and the node's own representation. Graph data is modelled and inferred by learning the interactions between nodes and the global structure of the graph.GNNs perform well in many graph related tasks such as node classification, link prediction and graph generation.

A typical graph neural network consists of multiple graph neural network layers, each of which consists of two main steps: information aggregation and feature update. Below is a simplified formulation of a graph neural network layer.

Information Aggregation Aggregation:

$h_i^{(l)} = AGGREGATE^{(l)}\left(\left\{h_j^{(l-l)}, \forall j \in \mathrm{Ne}(i)\right\}\right)$ denotes the hidden state of the node at layer i. $AGGREGATE^{(l)}$ is an aggregation function that aggregates the hidden states $h_j^{(l-l)}$ of the neighbours of node i, node j. $\mathrm{Ne}(i)$ denotes the set of neighbouring nodes connected to node i.

Feature Update (Update):

$h_{\mathrm{i}}^{(1)} = UPDATE^{(1)}\left(h_{\mathrm{i}}^{(1-1)}, h_{\mathrm{i}}^{(1)}\right)$ is an update function that combines the hidden state $h_{\mathrm{i}}^{(1-1)}$ of the previous layer of node i and the hidden state $h_{\mathrm{i}}^{(1)}$ of the current layer,to generate a new node representation.

By stacking multiple graph neural network layers, information can be propagated layer by layer from neighbouring nodes and capture relationships between nodes further away. Eventually, the graph neural network produces a final representation of each node that can be used for different graph-related tasks.

It is important to note that the above formulation is just a simple example, and there are many variants and improvements of graph neural network models in practice, such as GraphSAGE [21], GCN [22], GAT [23], etc., which may use different aggregation and updating functions, as well as other techniques to process graph data

2.3 Large Language Models

In recent years, researchers have paid more and more attention to the evolution and development of language models, by expanding the amount of pre-trained language models and the amount of data, Large language model can not only improve the effect of task processing, and can show many special capabilities that small models do not have.

The underlying layer of large language model basically adopts the Transformer structure. Currently, the common large language models are BERT [19], GPT [20]. BERT model adopts the bi-directional encoding layer in the 12-layer Transformer structure to represent the network, but it only adopts the encoding layer of the Transformer as the theme framework, which makes it more difficult to solve the tasks of text generation such as article continuation, translation, etc. GPT adopts the decoding layer of the Transformer as the theme framework, which makes it more difficult to solve the tasks of text generation such as article continuation and translation. GPT adopts the decoding layer of Transformer as the main structure of the network for modelling, so it is very effective in text generation tasks, compared with the BERT model, GPT does not care much about the understanding of the current language representation, and focuses on how to continue to generate the rest of the text.

Large language models also include other components and techniques such as positional coding, multilayer stacking, pre-training and fine-tuning. By pre-training on large-scale textual data, these models can learn rich linguistic knowledge and show strong performance capabilities in various natural language processing tasks.

2.4 Proposed Taxonomy

Depending on the final solution component for solving graph-related problems, we classify the combined LLM and GNN approach into three categories:(1) GNN as the final component for task solving. In this case, LLM functions as a text encoder, processing the input textual information to aid the GNN in task-solving. (2) LLM as the final component of task solving. In this category. In this type, there are two scenarios; one involves using GNNs to encode graph structures, thereby assisting the LLM in capturing graph structure information. The other involves transforming the graph structure into a sequence understandable by the LLM or adapting the transformer architecture to concurrently handle textual and graph structure information, thus eliminating the need for GNNs. (3) Collaborative solving of LLM and GNN. In this category, LLM and GNN co-solving can be done in two ways, either by co-training and sharing features, or by aligning the two through the latent space. In the next section, we will investigate and summarise each of these three categories individually. A model classification diagram and a representative example are given as shown in Fig. 1.

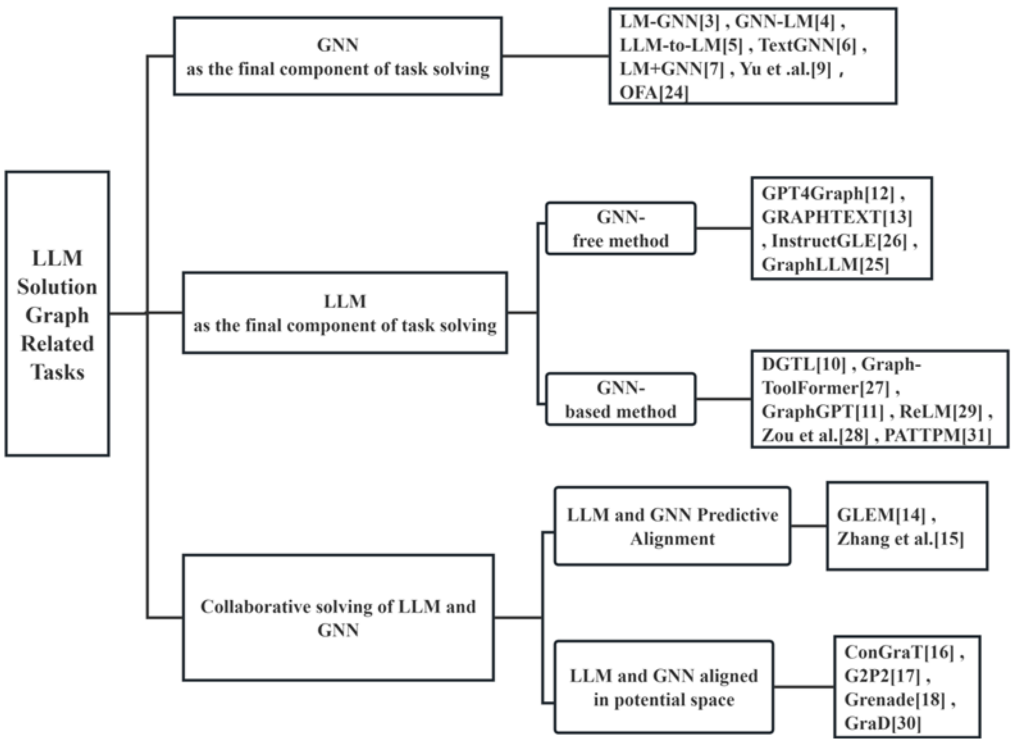
LLM Solution Graph Related Tasks
GNN as the final component of task solving
LM-GNN[3] , GNN-LM[4] , LLM-to-LM[5] , TextGNN[6] , LM+GNN[7] , Yu et .al.[9] , OFA[24]
LLM as the final component of task solving
GNN-free method
GPT4Graph[12] , GRAPHTEXT[13] , InstructGLE[26] , GraphLLM[25]
GNN-based method
DGTL[10] , Graph-ToolFormer[27] , GraphGPT[11] , ReLM[29] , Zou et al.[28] , PATTPM[31]
Collaborative solving of LLM and GNN
LLM and GNN Predictive Alignment
GLEM[14] , Zhang et al.[15]
LLM and GNN aligned in potential space
ConGraT[16] , G2P2[17] , Grenade[18] , GraD[30]

Figure 1: Classification and representative examples of models for solving graph-related tasks with the help of large language models (LLMs)

3. Fine Tuning

3.1 GNN as the final component of task solving

GNN performs well in areas such as processing graph-structured data, but has limitations in processing graphs with text, whereas LLM possesses a better ability to understand textual information. In this category, LLM acts as a related device for text feature extraction, providing initial node feature vectors to the GNN, which later generates node and edge representations and predictions through the GNN. In the next section, we will discuss the techniques related to these models.

GNNs excel in processing graph-structured data but often struggle with graphs containing textual elements. Conversely, LLMs demonstrate superior capabilities in understanding text. In this integration, the LLM serves as an auxiliary tool for text feature extraction, providing initial node feature vectors. These vectors are then further refined by the GNN, which generates comprehensive node and edge representations and predictions. The subsequent section will discuss the techniques related to these models in more detail.

LM-GNN [3] exemplifies the joint training of Large Language Models and Graph Neural Networks. Here, a graph-aware transformer functions as a semantic encoder, later fine-tuned in conjunction with a GNN encoder for predicting links in heterogeneous graphs. Meng et al. [4] introduced GNN-LM, a language modeling approach that enhances traditional neural network language models. It does so by referencing similar contexts across the entire training corpus, utilizing a high-dimensional tagged representation to retrieve the k-nearest neighbors of the input context as references. For each input context, a directed isomorphic graph is constructed, with nodes representing tokens from the input context or retrieved neighboring contexts and edges signifying connections between these tokens. A GNN then aggregates information from these contexts to decode the subsequent token. This methodology facilitates the retrieval of pertinent contexts as references, thereby improving the model's ability to predict forthcoming words in language modeling tasks.The LLM-to-LM [5] framework first wraps the textual attributes associated with each node in a custom prompt, and then uses the large language model to query and generate a list of predictions and explanations. Next, the raw text, predictions, and interpretations are used to fine-tune the language model and turned into vector node features. Finally, these node features can be used in a downstream graph neural network to predict the classes of unknown nodes.

TextGNN [6] integrates a text encoder with a Graph Neural Network (GNN), showcasing robust performance in tasks such as advertisement relevance. This model capitalizes on the text encoder's natural language understanding capabilities and enhances its performance by incorporating information from graph-type data, outperforming approaches that rely solely on semantic information. TextGNN employs an end-to-end framework that synergizes text encoders with Graph Neural Networks for training and optimization. Within this framework, the text encoder processes textual input, capturing its semantic essence, while the Graph Neural Network handles graph data, extracting graph relationships and contextual insights. Xie et al. [7] proposed a framework model for graph corpora, in which the framework LM+GNN consists mainly of one or more LMs are responsible for encoding textual information, while GNN aggregators are used for information aggregation. Given a graph corpus as input, LM+GNN uses one or more LMs as text encoders for the nodes. The embeddings generated through these LMs are added to the topology of the graph and fused with other information in the graph. Finally, the output is supervised by a task-specific decoder.

In comparison to Graph Neural Networks (GNNs), which often depend on high-quality labeling, Large Language Models (LLMs) boast an extensive knowledge base and exhibit remarkable zero-shot and few-shot learning capabilities. This is particularly evident in node classification tasks involving graphs with textual attributes. The integration of an LLM enables the model to effectively handle node classification tasks even with limited samples. Traditional GNNs typically require a substantial number of labeled samples to perform well, posing a challenge in scenarios with limited training data. However, LLMs, with their extensive pre-training and rich linguistic knowledge acquired from large-scale textual data, can mitigate this issue. When combined with a GNN, The LLM comprehends the semantic and contextual nuances of the text, facilitating the generation of high-quality node representations. LLM-GNN [8] is an unlabelled node classification method. The method combines the advantages of graph neural networks and large language models by using LLM to annotate a small number of nodes and training the GNN on the annotations of the large language model to predict the majority of unlabelled nodes. Yu et al. [9] proposed a method to enhance class-level information using Large Language Models to improve the quality of node representations, which was used to solve the problem of node classification tasks under a small number of samples. Semantic information is extracted from the labels using LLM and samples with labels are generated, whereas the structural information in the original dataset is later captured using an edge predictor and the newly generated samples are integrated into the original graph. Finally, the entire dataset is trained by graph neural network and the results of node classification are obtained. OFA [24] describes different graph data through natural language and introduces the concept of nodes of interest, uses a single task to standardise different tasks, and converts all inputs embedded in llm into cued graphs containing both graph and task information through a graphical cueing paradigm, thus allowing adaptive downstream prediction. Experimentally, OFA was found to be capable of under-shooting and zero-shooting learning on different graph domains.

3.2 LLM as the final component of task solving

The core concept of this category centers on employing Large Language Models (LLMs) as the primary architectural framework to acquire both graph structure and textual information. Given that graphs vary in structure and feature different forms of definitions, transforming graph data directly into text is not straightforward, posing a significant challenge for the application of LLMs to graph-related tasks. This category can be further subdivided based on whether Graph Neural Networks (GNNs) are involved in the task-solving process. Accordingly, this section is divided into two subcategories: GNN-free method and GNN-based methods

3.2.1 GNN-free methods

This type of approach uses LLM directly to obtain textual information and graph structure without GNN involvement. Traditional LLMs use transformers for natural language encoding, but have limited ability to model graph structure information. Therefore, this type of approach obtains node and edge representations by converting the graph structure into textual information or by designing the LLM as an advanced modelling structure capable of processing textual information and encoding it graphically. GPT4Graph [12] converts graph data into a graphical Description Language. GRAPHTEXT [13] encodes graph information into text sequences. InstructGLM [26] uses natural language to describe the geometric structure and node characteristics of graphs, and enables LLM to solve graph-related problems by tuning it with instructions. The above method enables LLM to process graph data directly by converting graph data into textual descriptions, but LLM needs to identify the implicit graph structure from sequential text in the process, and compared with traditional graph learning methods, LLM may face inefficiencies in graph learning based on sequential graph descriptions, and is still insufficient for representing multidimensional and correlated graph data.

GraphLLM [25] is an end-to-end approach that synergistically integrates a graph learning model (graph converter) with an LLM into a single system, with a framework that consists of three main steps: node understanding, structural understanding, and LLM-oriented prefix tuning for graph enhancement. Compared to methods that convert graph data to text, GraphLLM is able to improve on graph reasoning tasks by exploiting synergy with graph converters and leveraging the strengths of both.

3.2.2 GNN-based method

The ability of GNNs to capture hidden representations of structural information between nodes when processing structured data provides a powerful representation learning capability. This has led to the utilisation of GNNs in a number of approaches to study LLMs when processing graph data to enhance the performance of LLMs.

DGTL [10] utilises large language models to provide prediction and achieve interpretability for text-attributed graph related tasks, the framework combines a disentangled graph learning The framework combines the method of untangled graph learning to generate text embeddings by computing the average of the last layer of features in the upstream DGTL, capturing the contextual and semantic information of the text associated with each node, and then using the untangled graph learning to learn embeddings with different domain information, and then finally injecting the learnt features with domain information into the downstream DGTL.The LLM and GNN in the Graph-ToolFormer [27] framework are individually pre-trained, and then the LLM calls the pre-trained GNN model to complete the task. That is, LLM is used as a unified common interface for graph inference tasks.

GraphGPT [11] enhances the understanding and adaptation of graph structures by aligning them to the natural language space and through graph instruction tuning. In addition, in order to improve the stepwise reasoning ability of large language models, GraphGPT also integrates thought chain distillation into the framework, which makes the whole model show stronger ability in stepwise reasoning and handling distributed transfer.ReLM [29] makes use of LM and GNN for chemical reaction prediction, using pre-trained GNNs to generate candidate answers and context examples from a pool of candidates, which are then analysed in a multiple-choice format using LM.

Zou et al. [28] proposed a new pre-training framework for topology perception by jointly optimising LM and graph neural networks to predict the nodes involved in the context graph. In addition, based on the situation that some nodes are rich in textual information while others have less textual information, an enhancement strategy is designed to enrich the nodes with text from neighbouring nodes with insufficient textual information. After finishing the pre-training, only the LM is applied to the downstream task and the auxiliary role of the GNN is abandoned. PATTON [31] utilises textual information and network structure to enhance and consolidate the LM's ability to comprehend tokens and documents. The GNN nested Transformer architecture GraphFormers proposed by Yang et al. [32] is used in the framework, while two pre-training strategies are later employed to help the LM capture the intrinsic dependencies that exist between textual attributes and network structures.

3.3 Collaborative solving of LLM and GNN

These studies combine GNN for graph structure coding and LLM for textual information coding for co-training and mutual enhancement.The GNN component can provide structural information to the whole framework and provide it to LLM, and LLM can provide textual analysing capability to the whole framework and provide textual signals to GNN. Depending on how the two combine and learn from each other, we divide this category into two types: the LLM and GNN predictive alignment, and the LLM and GNN alignment in potential space.

3.3.1 LLM and GNN Predictive Alignment

GLEM [14] makes use of the relevant definitions of the Variational EM framework, where the large language model uses the textual information of each node to predict its labels and to model the distribution of labels based on local textual attributes. While graph neural networks use the text and label information of the surrounding nodes to make label predictions and represent the label distribution under global conditions.GLEM makes the language model and graph neural networks collaborate with each other by alternating the optimisation of the E-step and the M-step. Specifically, in the E-step, the graph neural network is fixed and the language model is made to mimic the label inference of the GNN in order to transfer the global knowledge learnt by the GNN to the LM.In the M-step, the LM is fixed and the node representations learnt by the LM are used as features for label prediction by optimising the GNN. The alternating training of the two steps enables the GNN to effectively capture the global correlation of nodes and thus achieve accurate label prediction.

Zhang et al. [15] propose a co-training approach that enables classification and pseudo-labelling of textual attribute maps by combining a text analysis module and a network learning module. The framework models both the original text and the network structure, and enhances both modules by co-training and feature sharing.

3.3.2 LLM and GNN aligned in potential space

ConGraT [16] jointly learns graph nodes and text representations by using two independent encoders that are aligned in a common latent space and training. This approach receives inspiration from previous work in the area of joint text and image coding, and extends the training objective to take into account node similarity and reasonably guessed information.

The G2P2 [17] methodology study utilises a converter-based text encoder and a GNN-based graphical encoder to improve text classification performance. In this case, the converter is used as a text encoder and on the other hand, the GNN serves as a graph encoder taking the graph kernel node features as input and generating node embedding vectors for each node. By combining the coding capabilities of the converter and the GNN, the framework is able to provide more comprehensive node representations, and these node embedding vectors contain both textual information and information about the graph structure, thus better capturing the semantic and associative relationships between nodes.

Grenade [18] optimises self-supervised learning algorithms in graphs to capture both textual semantic and structural contextual information. Grenade exploits the synergistic effects of pre-trained language models and graph neural networks, and jointly optimises two self-supervised learning algorithms, graph-centric comparison learning and graph-centric knowledge alignment.

GraD [30] encodes graph structures into LMs for fast inference without graphs, and joint training of teacher GNNs and students without graphs through shared LMs allows the two models to learn from each other and improve overall performance.

4. Challenges and future directions

While the preceding sections have outlined the current landscape of using language models in graph learning, there remains significant potential for further research in this domain. In this section, we briefly examine some limitations of language models when applied to graph learning and suggest potential directions for future research.

Lack of effective pre-training algorithms. most current language models are based on self-supervised pre-training, but this approach is not effective in graph learning. Therefore, it remains a challenge to effectively pre-train on large-scale graph data. Exploring pre-training methods on graphs is a valuable research direction.

Insufficient ability to represent structural information. since the training is mainly based on textual data, language models lack in grasping the complexity of graph structural information, and generating topology-based supervised signals using language models is a challenging problem. A way to address this problem could be by designing specific pre-training goals to guide the language model to learn the representation of topological structures.

5. Conclusion

The application of Large Language Models (LLMs) to graph-related tasks has emerged as a vital research area in recent years. To classify and provide a comprehensive overview of this field, we propose a novel classification method. This method categorizes techniques involving graphs and textual information into three distinct categories:LLMs as the final component of task solving, GNNs as the final component of task solving, and collaborative solving of LLM and GNN. Based on this categorisation, we systematically review representative studies and discuss some limitations and future research directions in this direction. It is hoped that this comprehensive review will reveal the potential of LLM in the field of graph learning, as well as the advances and challenges made, and provide insights for further developments in the field.

References

[1] J. Liu, C. Yang, Z. Lu, J. Chen, Y. Li, M. Zhang, et al., “Towards graph foundation models: A survey and beyond,” arXiv preprint arXiv: 2310. 11829, 2023.
[2] Y. Li, Z. Li, P. Wang, J. Li, X. Sun, H. Cheng, et al., “A survey of graph meets large language model: Progress and future directions,” arXiv preprint arXiv: 2311. 12399, 2023.
[3] V. N. Ioannidis, X. Song, D. Zheng, H. Zhang, J. Ma, Y. Xu, et al., “Efficient and effective training of language and graph neural network models,” arXiv preprint arXiv: 2206. 10781, 2022.
[4] Y. Meng, S. Zong, X. Li, X. Sun, T. Zhang, F. Wu, et al., “GNN-LM: Language Modeling based on Global Contexts via GNN,” arXiv preprint arXiv: 2110. 08743, 2021.
[5] X. He, X. Bresson, T. Laurent, et al., “Harnessing Explanations: LLM-to-LM Interpreter for Enhanced Text-Attributed Graph Representation Learning,” arXiv preprint arXiv: 2305. 19523, 2023.
[6] J. Y. Zhu, Y. Cui, Y. Liu, H. Sun, X. Li, M. Pelger, et al., “TextGNN: Improving Text Encoder via Graph Neural Network in Sponsored Search,” Proceedings of the Web Conference 2021, pp. 2848-2857, 2021.
[7] H. Xie, D. Zheng, J. Ma, H. Zhang, V. N. Ioannidis, X. Song, et al., “Graph-Aware Language Model PreTraining on a Large Graph Corpus Can Help Multiple Graph Applications,” Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023.
[8] Z. Chen, H. Mao, H. Wen, H. Han, W. Jin, H. Zhang, et al., “Label-free node classification on graphs with large language models (llms),” arXiv preprint arXiv: 2310. 04668, 2023.
[9] J. Yu, Y. Ren, C. Gong, J. Tan, X. Li, X. Zhang, “Empower Text-Attributed Graphs Learning with Large Language Models,” arXiv preprint arXiv: 2310. 09872, 2023.
[10] Y. Qin, X. Wang, Z. Zhang, W. Zhu, “Disentangled Representation Learning with Large Language Models for Text-Attributed Graphs,” arXiv preprint arXiv: 2310. 18152, 2023.
[11] J. Tang, Y. Yang, W. Wei, L. Shi, L. Su, S. Cheng, et al., “GraphGPT: Graph Instruction Tuning for Large Language Models,” arXiv preprint arXiv: 2310. 13023, 2023.

[12] J. Guo, L. Du, H. Liu, “GPT4Graph: Can Large Language Models Understand Graph Structured Data? An Empirical Evaluation and Benchmarking,” arXiv preprint arXiv: 2305. 15066, 2023.
[13] J. Zhao, L. Zhuo, Y. Shen, M. Qu, K. Liu, M. Bronstein, et al., “GraphText: Graph Reasoning in Text Space,” arXiv preprint arXiv: 2310. 01089, 2023.
[14] J. Zhao, M. Qu, C. Li, H. Yan, Q. Liu, R. Li, et al., “Learning on Large-scale Text-attributed Graphs via Variational Inference,” arXiv preprint arXiv: 2210. 14709, 2022.
[15] X. Zhang, C. Zhang, X. L. Dong, J. Shang, J. Han, “Minimally-supervised structure-rich text categorization via learning on text-rich networks,” Proceedings of the Web Conference 2021, pp. 3258-3268, 2021.
[16] W. Brannon, S. Fulay, H. Jiang, W. Kang, B. Roy, J. Kabbara, et al., “ConGraT: Self-Supervised Contrastive Pretraining for Joint Graph and Text Embeddings,” arXiv preprint arXiv: 2305. 14321, 2023.
[17] Z. Wen, Y. Fang, “Augmenting Low-Resource Text Classification with Graph-Grounded Pre-training and Prompting,” arXiv preprint arXiv: 2305. 03324, 2023.
[18] Y. Li, K. Ding, K. Lee, “GRENADE: GraphCentric Language Model for Self-Supervised Representation Learning on Text-Attributed Graphs,” arXiv preprint arXiv: 2310. 15109, 2023.
[19] J. Devlin, M. W. Chang, K. Lee, et al., “Bert: Pretraining of deep bidirectional transformers for language understanding,” arXiv preprint arXiv: 1810. 04805, 2018.
[20] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, et al., “Sparks of artificial general intelligence: Early experiments with gpt-4,” arXiv preprint arXiv: 2303. 12712, 2023.
[21] W. Hamilton, Z. Ying, J. Leskovec, “GRENADE: GraphCentric Language Model for Self-Supervised Representation Learning on Text-Attributed Graphs,” Advances in neural information processing systems, vol. 30, 2017.
[22] T. N. Kipf, M. Welling, “Semi-supervised classification with graph convolutional networks,” arXiv preprint arXiv: 1609. 02907, 2016.
[23] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, “Graph attention networks,” arXiv preprint arXiv: 1710. 10903, 2017.
[24] H. Liu, J. Feng, L. Kong, N. Liang, D. Tao, Y. Chen, et al., “One for All: Towards Training One Graph Model for All Classification Tasks,” arXiv preprint arXiv: 2310. 00149, 2023.
[25] Z. Chai, T. Zhang, L. Wu, K. Han, X. Hu, X. Huang, et al., “Graphllm: Boosting graph reasoning ability of large language mode,” arXiv preprint arXiv: 2310. 05845, 2023.
[26] R. Ye, C. Zhang, R. Wang, S. Xu, Y. Zhang, “Natural Language is All a Graph Needs,” arXiv preprint arXiv: 2308. 07134, 2023.
[27] J. Zhang, “Graph-ToolFormer: To Empower LLMs with Graph Reasoning Ability via Prompt Augmented by ChatGPT,” arXiv preprint arXiv: 2304. 11116, 2023.
[28] T. Zou, L. Yu, Y. Huang, L. Sun, B. Du, “Pretraining Language Models with Text-Attributed Heterogeneous Graphs,” arXiv preprint arXiv: 2310. 12580, 2023.
[29] Y. Shi, A. Zhang, E. Zhang, Z. Liu, X. Wang, “Relm: Leveraging language models for enhanced chemical reaction prediction,” arXiv preprint arXiv: 2310. 13590, 2023.
[30] C. Mavromatis, V. N. Ioannidis, S. Wang, D. Zheng, S. Adeshina, J. Ma, et al., “Train your own gnn teacher: Graph-aware distillation on textual graphs,” arXiv preprint arXiv: 2304. 10668, 2023.
[31] B. Jin, W. Zhang, Y. Zhang, Y. Meng, X. Zhang, Q. Zhu, et al., “Patton: Language model pretraining on text-rich networks,” arXiv preprint arXiv: 2305. 12268, 2023.

[32] J. Yang, Z. Liu, S. Xiao, C. Li, D. Lian, S. Agrawal, et al., "GraphFormers: GNN-nested transformers for representation learning on textual graph," 35th Conference on Neural Information Processing Systems, vol. 34, pp. 28798-28810, 2021