Large Language Models on Graphs: A Comprehensive Survey

Bowen Jin, Gang Liu, *Graduate Student Member, IEEE*, Chi Han, Meng Jiang, Heng Ji, *Member, IEEE*, and Jiawei Han, *Fellow, IEEE*

(Survey Paper)

***Abstract*—Large language models (LLMs), such as GPT4 and LLaMA, are creating significant advancements in natural language processing, due to their strong text encoding/decoding ability and newly found emergent capability (e.g., reasoning). While LLMs are mainly designed to process pure texts, there are many real-world scenarios where text data is associated with rich structure information in the form of graphs (e.g., academic networks, and e-commerce networks) or scenarios where graph data is paired with rich textual information (e.g., molecules with descriptions). Besides, although LLMs have shown their pure text-based reasoning ability, it is underexplored whether such ability can be generalized to graphs (i.e., graph-based reasoning). In this paper, we provide a systematic review of scenarios and techniques related to large language models on graphs. We first summarize potential scenarios of adopting LLMs on graphs into three categories, namely pure graphs, text-attributed graphs, and text-paired graphs. We then discuss detailed techniques for utilizing LLMs on graphs, including LLM as Predictor, LLM as Encoder, and LLM as Aligner, and compare the advantages and disadvantages of different schools of models. Furthermore, we discuss the real-world applications of such methods and summarize open-source codes and benchmark datasets. Finally, we conclude with potential future research directions in this fast-growing field.**

***Index Terms*—Graph neural networks, graph representation learning, large language models (LLMs), natural language processing.**

Received 1 February 2024; revised 16 June 2024; accepted 10 September 2024. Date of publication 27 September 2024; date of current version 13 November 2024. This work was supported in part by US DARPA INCAS under Program HR0011-21-C0165, in part by BRIES under Program HR0011-24-3-0325, in part by National Science Foundation under Grant IIS-19-56151, in part by the Molecule Maker LabInstitute: An AI Research Institutes program supported by NSF under Award 2019897, in part by the Institute for Geospatial Understanding through an Integrative Discovery Environment (I-GUIDE) by NSF under Award 2118329, in part by U.S. DARPA ITM under Program FA8650-23-C-7316, in part by Agriculture and Food Research Initiative (AFRI) under Grant 2020-67021-32799/project accession no. 1024178 from the USDA National Institute of Food and Agriculture, in part by NSF under Award 2142827, Award 2146761, and Award 2234058, and in part by ONR under Grant N00014-22-1-2507. Recommended for acceptance by K. Zheng. *(Bowen Jin, Gang Liu, and Chi Han contributed equally to this work.) (Corresponding author: Bowen Jin.)*

Bowen Jin, Chi Han, Heng Ji, and Jiawei Han are with the University of Illinois at Urbana-Champaign, Champaign, IL 61820 USA (e-mail: bowenj4@illinois.edu; chihan3@illinois.edu; hengji@illinois.edu; hanj@illinois.edu).

Gang Liu and Meng Jiang are with the University of Notre Dame, Notre Dame, IN 46556 USA (e-mail: gliu7@nd.edu; mjiang2@nd.edu).

The related source can be found at https://github.com/PeterGriffinJin/Awesome-Language-Model-on-Graphs.

This article has supplementary downloadable material available at https://doi.org/10.1109/TKDE.2024.3469578, provided by the authors.

Digital Object Identifier 10.1109/TKDE.2024.3469578

I. Introduction

LARGE language models (LLMs) (e.g., BERT [23], T5 [29], LLaMA [118]) which represents a direction of ever-increasing models' sizes pre-trained on larger corpora, have demonstrated powerful capabilities in solving natural language processing (NLP) tasks, including question answering [1], text generation [2] and document understanding [3]. There are no clear and static thresholds regarding the model sizes. Early LLMs (e.g., BERT [23], RoBERTa [24]) adopt an encoder-only architecture and show capabilities in text representation learning [4] and natural language understanding [3]. In recent years, more focus has been given to larger decoder-only architectures [118] or encoder-decoder architectures [29]. As the model size scales up, such LLMs have also shown reasoning ability and even more advanced emergent ability [5], exposing a strong potential for Artificial General Intelligence (AGI).

While LLMs are extensively applied to process pure texts, there is an increasing number of applications where the text data are associated with structure information which are represented in the form of graphs. As presented in Fig. 1, in academic networks, papers (with title and description) and authors (with profile text), are interconnected with authorship relationships. Understanding both the author/paper's text information and author-paper structure information on such graphs can contribute to advanced author/paper modeling and accurate recommendations for collaboration; In the scientific domain, molecules are represented as graphs and are often paired with text that describes their basic properties (e.g., mass and weight). Joint modeling of both the molecule structure (graph) and the associated rich knowledge (text) is important for deeper molecule understanding. Since LLMs are mainly proposed for modeling texts that lie in a sequential fashion, those scenarios mentioned above pose new challenges on how to enable LLMs to encode the structure information on graphs. In addition, since LLMs have demonstrated their superb text-based reasoning ability, it is promising to explore whether they have the potential to address fundamental graph reasoning problems on pure graphs. These graph reasoning tasks include inferring connectivity [6], shortest path [7], subgraph matching [8], and logical rule induction [18].

Recently, there has been an increasing interest [9] in extending LLMs for graph-based applications (summarized in Fig. 1). According to the relationship between graph and text presented

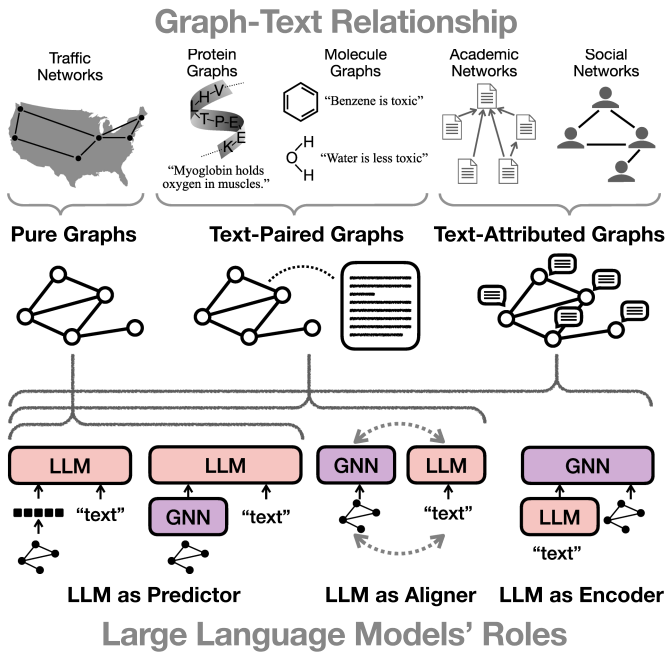


Fig. 1. According to the relationship between graph and text, we categorize three LLM on graph scenarios. Depending on the role of LLM, we summarize three LLM-on-graph techniques. "LLM as Predictor" is where LLMs are responsible for predicting the final answer. "LLM as Aligner" will align the inputs-output pairs with those of GNNs. "LLM as Encoder" refers to using LLMs to encode and obtain feature vectors.

in Fig. 1, the application scenarios can be categorized into pure graphs, text-attributed graphs (nodes/edges are associated with texts), and text-paired graphs. Depending on the role of LLMs and their interaction with graph neural networks (GNNs), the LLM on graphs techniques can be classified into treating LLMs as the final component for prediction (LLM as Predictor), treating LLMs as the feature extractor for GNNs (LLM as Encoder), and align the latent space of LLMs with GNNs (LLM as Aligner).

There are a limited number of existing surveys exploring the intersection between LLMs and graphs. Related to deep learning on graphs, Liu et al. [20] discuss pretrained foundation models on graphs, including their backbone architectures, pretraining methods, and adaptation techniques. Pan et al. [21] review the connection between LLMs and knowledge graphs (KGs) especially on how KGs can enhance LLMs training and inference, and how LLMs can facilitate KG construction and reasoning. Mao et al. [203] and Li et al. [204] review LLM on graphs focusing on techniques rather than applications. In summary, existing surveys either focus more on GNNs rather than LLMs or fail to provide a systematic perspective on their applications in various graph scenarios as in Fig. 1. Our paper provides a comprehensive review of the LLMs on graphs for broader researchers from diverse backgrounds besides the computer science and machine learning community who want to enter this rapidly developing field (Fig. 2).

Our Contributions: The notable contributions of our paper are summarized as follows:

- *Categorization of Graph Scenarios:* We systematically summarize the graph scenarios where language models can be adopted into: pure graphs, text-attributed graphs, and text-paired graphs.
- *Systematic Review of Techniques:* We provide the most comprehensive overview of language models on graph techniques. For different graph scenarios, we summarize the representative models, provide detailed illustrations of each of them, and make necessary comparisons.
- *Abundant Resources:* We collect abundant resources on language models on graphs, including benchmark datasets, open-source codebases, and practical applications.
- *Future Directions:* We delve into the foundational principles of language models on graphs and propose six prospective avenues for future exploration.

Organization of Survey: The rest of this survey is organized as follows. Section II-B introduces the background of LLMs and GNNs, lists commonly used notations, and defines related concepts. Section III categorizes graph scenarios where LLMs can be adopted and summarizes LLMs on graph techniques. Sections IV, V, and VI provides a detailed illustration of LLM methodologies for different graph scenarios. Section VII delivers available datasets, open-source codebases, and a collection of applications across various domains. Section VIII introduces some potential future directions. Section IX summarizes the paper.

II. Definitions & Background

A. Definitions

We provide definitions of various types of graphs and introduce the notations (as shown in Table I) in this section.

Definition 1 (Graph): A graph can be defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Here $\mathcal{V}$ signifies the set of nodes, while E denotes the set of edges. A specific node can be represented by $v_i \in \mathcal{V}$, and an edge directed from node v_j to v_i can be expressed as $e_{ij} = (v_i, v_j) \in \mathcal{E}$. The set of nodes adjacent to a particular node v is articulated as $N(v) = \{u \in \mathcal{V} | (v, u) \in \mathcal{E}\}$.

Definition 2 (Graph with node-level textual information): This type of graph can be denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{D})$, where $\mathcal{V}$, $\mathcal{E}$ and $\mathcal{D}$ are node set, edge set, and text set, respectively. Each $v_i \in \mathcal{V}$ is associated with some textual information $d_{v_i} \in \mathcal{D}$. For instance, in an academic citation network, one can interpret $v \in \mathcal{V}$ as the scholarly articles, $e \in \mathcal{E}$ as the citation links between them, and $d \in \mathcal{D}$ as the textual content of these articles. A graph with node-level textual information is also called a **text-attributed graph** [31], a text-rich graph [61], or a textual graph [71].

Definition 3 (Graph with edge-level textual information): This type of graph can be denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{D})$. Each $e_{ij} \in \mathcal{E}$ is associated with some textual information $d_{e_{ij}} \in \mathcal{D}$. For example, in a social network, one can interpret $v \in \mathcal{V}$ as the users, $e \in \mathcal{E}$ as the interaction between the users, and $d \in \mathcal{D}$ as the textual content of the messages sent between the users. Such a graph is also called a textual-edge network [73].

Definition 4 (Graph with graph-level textual information): This type of graph can be denoted as the pair $(\mathcal{G}, d_{\mathcal{G}})$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. $\mathcal{V}$ and $\mathcal{E}$ are node set and edge set. $d_{\mathcal{G}}$ is the text

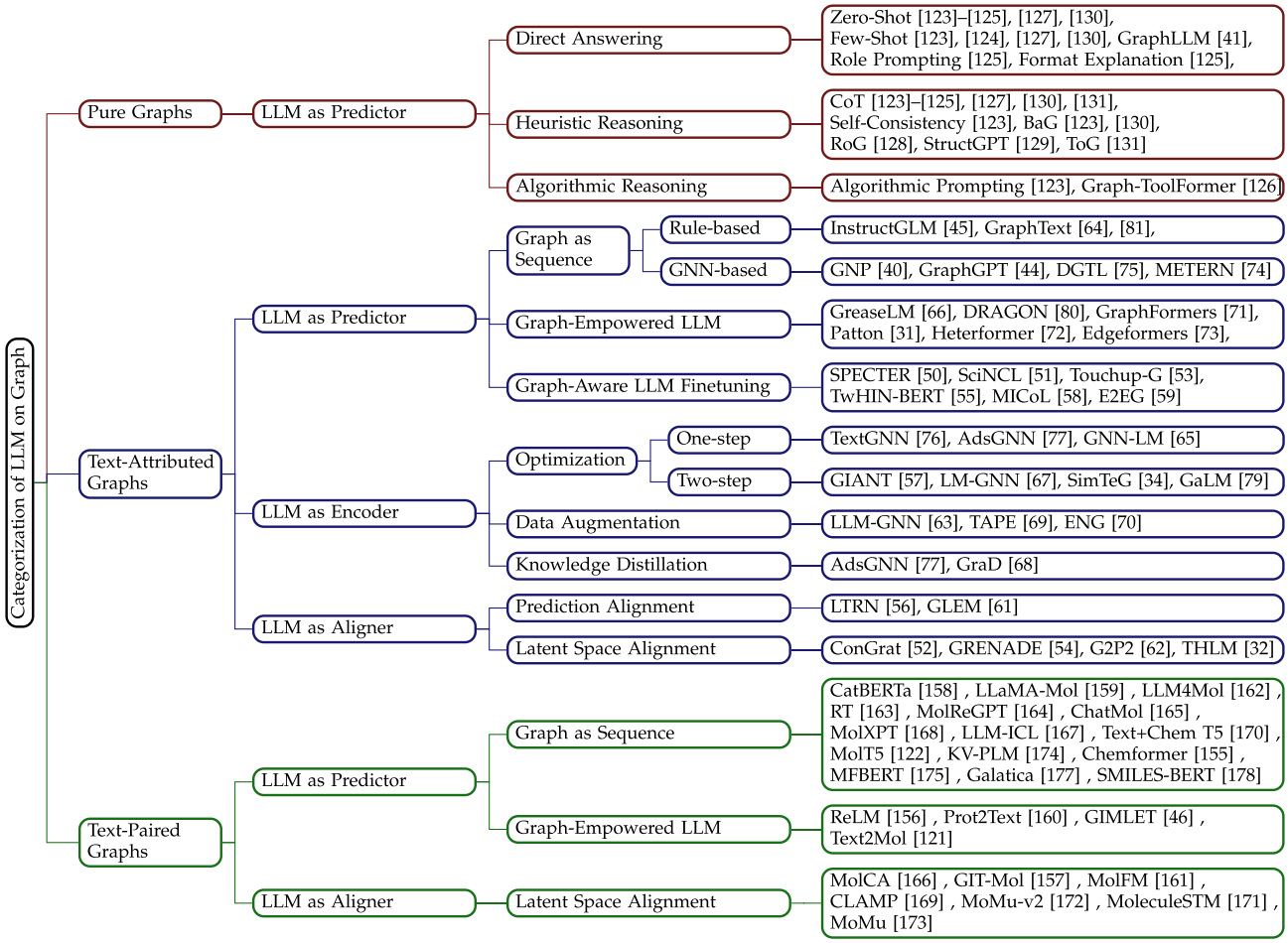


Fig. 2. A taxonomy of LLM on graph scenarios and techniques with representative examples.

set paired to the graph $\mathcal{G}$. For instance, in a molecular graph $\mathcal{G}$, $v \in \mathcal{V}$ denotes an atom, $e \in \mathcal{E}$ represents the strong attractive forces or chemical bonds that hold molecules together, and $d_{\mathcal{G}}$ represents the textual description of the molecule. We note that texts may also be associated with subgraph-level concepts and then paired with the entire graph. Such a graph is also called a **text-paired graph**.

B. Background

(Large) Language Models: Language Models (LMs), or language modeling, is an area in the field of natural language processing (NLP) on understanding and generation from text distributions. In recent years, large language models (LLMs) have demonstrated impressive capabilities in tasks such as machine translation, text summarization, reasoning, and question answering [26], [42], [111], [112], [113], [114], [194], [208].

Language models have evolved significantly over time. BERT [23] marks significant progress in language modeling and representation. BERT models the conditional probability of a word given its bidirectional context, also named masked language modeling (MLM) objective :

$$\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}\left[\sum_{s_i\in\mathcal{S}}\log p(s_i|s_1,\ldots,s_{i-1},s_{i+1},\ldots,s_{N_{\mathcal{S}}})\right], \quad (1)$$

where $\mathcal{S}$ is a sentence sampled from the corpus $\mathcal{D}$, s_i is the i-th word in the sentence, and $N_{\mathcal{S}}$ is the length of the sentence. On the other hand, the objective of causal language modeling or text generation is defined as:

$$\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}\left[\sum_{s_i\in\mathcal{S}}\log p(s_i|s_1,\ldots,s_{i-1})\right]. \quad (2)$$

Following BERT, other masked language models are proposed, such as RoBERTa [24], ALBERT [115], and ELECTRA [116], with similar architectures and objectives of text representation. Efforts have been made to combine language models with other modalities such as vision [95], [120] and biochemical structures [46], [121], [122]. In this paper, we will discuss its combination with graphs.

The lifecycle of an LLM usually involves some or all the following steps: pretraining, finetuning, and prompting. In pretraining, LLMs are usually trained on a larger corpus with multiple language modeling objectives [23], [26], [28], which

TABLE I
NOTATIONS OF CONCEPTS

Notations	Descriptions
$\|\cdot\|$	The length of a set.
$[\mathbf{A}, \mathbf{B}]$	The concatenation of $\mathbf{A}$ and $\mathbf{B}$.
$\|\|$	Concatenate operation.
$\mathcal{G}$	A graph.
$\mathcal{V}$	The set of nodes in a graph.
v	A node $v \in \mathcal{V}$.
$\mathcal{E}$	The set of edges in a graph.
e	An edge $e \in \mathcal{E}$.
$\mathcal{G}_v$	The ego graph associated with v in $\mathcal{G}$.
$N(v)$	The neighbors of a node v.
M	A meta-path or a meta-graph.
$N_M(v)$	The nodes which are reachable from node v with meta-path or meta-graph M.
$\mathcal{D}$	The text set.
$s \in \mathcal{S}$	The text token in a text sentence $\mathcal{S}$.
d_{v_i}	The text associated with the node v_i.
$d_{e_{ij}}$	The text associated with the edge e_{ij}.
$d_{\mathcal{G}}$	The text associated with the graph $\mathcal{G}$.
n	The number of nodes, $n = \|V\|$.
b	The dimension of a node hidden state.
$\mathbf{x}_{v_i} \in \mathbf{R}^d$	The initial feature vector of the node v_i.
$\mathbf{H}_v \in \mathbf{R}^{n \times b}$	The node hidden feature matrix.
$\mathbf{h}_{v_i} \in \mathbf{R}^b$	The hidden representation of node v_i.
$\mathbf{h}_{\mathcal{G}} \in \mathbf{R}^b$	The hidden representation of a graph $\mathcal{G}$.
$\mathbf{h}_{d_v} \in \mathbf{R}^b$	The representation of text d_v.
$\mathbf{H}_{d_v} \in \mathbf{R}^{\|d_v\| \times b}$	The hidden states of tokens in d_v.
$\mathbf{W}, \mathbf{\Theta}, w, \theta$	Learnable model parameters.
LLM$(\cdot)$	Large Language model.
GNN$(\cdot)$	Graph neural network.

aims to endow LLMs with strong language understanding and completion capability. If domain-specific abilities are expected, LLMs are then finetuned with a smaller amount of domain-specific data [36], [37], [38], [39], [42], [43]. Human preference optimization methods are sometimes applied after this stage to align outputs better with users' intentions or social values [205], [206], [207]. Finally, various prompting or prompt engineering techniques can be deployed to boost downstream task performance [47], [48], [49]. A more comprehensive description can be found in Appendix A, available online

We would like to point out that the word "large" in LLM is not associated with a clear and static threshold to divide language models. "Large" actually refers to a direction in which language models are inevitably evolving, and larger foundational models tend to possess significantly more representation and generalization power. Hence, we define LLMs to encompass both medium-scale PLMs, such as BERT, and large-scale LMs, like GPT-4, as suggested by [21].

Graph Neural Networks & Graph Transformers: In real-world scenarios, not all the data are sequential like text, many data lies in a more complex non-euclidean structure, i.e., graphs. GNN is proposed as a deep-learning architecture for graph data. Primary GNNs including GCN [83], GraphSAGE [84] and, GAT [85] are designed for solving node-level tasks. They mainly adopt a propagation-aggregation paradigm to obtain node representations:

$$\boldsymbol{h}_v^{(l)} = \text{AGG}^{(l)}\left(\boldsymbol{h}_v^{(l-1)}, \text{PROP}^{(l)}\left(\{\boldsymbol{h}_u^{(l-1)} \mid u \in \mathcal{N}(v)\}\right)\right).$$

When propagation is global ($u \in \mathcal{V}$), the Graph Transformer [140], [141] with attention-weighted node importance during sum aggregation can be defined. Let $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$ be the query, key, and value matrices, respectively, and $k_{\exp}$ denote the similarity between two nodes. Then, we have:

$$\text{Attn}(\mathbf{h}_v^{(l-1)}) = \sum_{u \in \mathcal{V}} \frac{k_{\exp}(\mathbf{h}_v^{(l-1)}, \mathbf{h}_u^{(l-1)})}{\sum_{w \in \mathcal{V}} k_{\exp}(\mathbf{h}_v^{(l-1)}, \mathbf{h}_w^{(l-1)})} \mathbf{h}_u^{(l-1)} \mathbf{W}_V,$$

where $k_{\exp}(\mathbf{h}_v^{(l-1)}, \mathbf{h}_w^{(l)}) = \exp(\frac{\mathbf{h}_v^{(l-1)} \mathbf{W}_Q \mathbf{h}_w^{(l-1)} \mathbf{W}_K}{\sqrt{d_K}})$.

To solve graph-level tasks, GNN models like GIN [188] or Graph Transformers obtain graph representations using a READOUT function: $\boldsymbol{h}_{\mathcal{G}} = \text{READOUT}(\{\boldsymbol{h}_{v_i} \mid v_i \in \mathcal{G}\})$. The READOUT functions include mean pooling, max pooling, and so on. Subsequent work on GNN tackles the issues of over-smoothing [138], over-squashing [139], interpretability [144], and bias [142]. While message-passing-based GNNs excel in structure encoding, researchers aim to enhance their expressiveness with Graph Transformers. These models leverage global multi-head attention mechanisms and integrate graph inductive biases through positional encoding, structural encoding, combining message-passing with attention layers, or improving attention efficiency on large graphs. Graph Transformers have been proven to be a state-of-the-art solution for many pure graph problems.

Language Models Versus Graph Transformers: Modern language models and graph Transformers both use Transformers [92] as the base model architecture. This makes the two concepts hard to distinguish, especially when the language models are adopted on graph applications. In this paper, "Transformers" typically refers to Transformer language models for simplicity. Here, we provide three points to help distinguish them: 1) *Tokens* (word token versus node token): Transformers take a token sequence as inputs. For language models, the tokens are word tokens; while for graph Transformers, the tokens are node tokens. In those cases where tokens include both word tokens and node tokens if the backbone Transformers is pretrained on text corpus (e.g., BERT [23] and LLaMA [118]), we will call it a "language model". 2) *Positional Encoding* (sequence versus graph): language models typically adopt the absolute or relative positional encoding considering the position of the word token in the sequence, while graph Transformers adopt shortest path distance [140], random walk distance, the eigenvalues of the graph Laplacian [141] to consider the distance of nodes in the graph. 3) *Goal* (text versus graph): The language models are originally proposed for text encoding and generation; while graph Transformers are proposed for node encoding or graph encoding. In those cases where texts are served as nodes/edges on the graph if the backbone Transformers is pretrained on text corpus, we will call it a "language model".

III. CATEGORIZATION AND FRAMEWORK

In this section, we first introduce our categorization of graph scenarios where language models can be adopted. Then we discuss the categorization of LLM on graph techniques. Finally,

we summarize the training & inference framework for language models on graphs.

A. Categorization of Graph Scenarios With LLMs

Pure Graphs without Textual Information are graphs with no text information or no semantically rich text information. Examples include traffic graphs and power transmission graphs. Those graphs often serve as context to test the graph reasoning ability of large language models (solve graph theory problems) or serve as knowledge sources to enhance the large language models (alleviate hallucination).

Text-Attributed Graphs refer to graphs where nodes or edges are associated with semantically rich text information. They are also called text-rich networks [31], textual graphs [71] or textual-edge networks [73]. Examples include academic networks, e-commerce networks, social networks, and legal case networks. On these graphs, researchers are interested in learning representations for nodes or edges with both textual and structure information [71][73].

Text-Paired Graphs have textual descriptions defined for the entire graph structure. For example, graphs like molecules may be paired with captions or textual features. While the graph structure significantly contributes to molecular properties, text descriptions can complement our understanding of molecules. The graph scenarios can be found in Fig. 1.

B. Categorization of LLMs on Graph Techniques

According to the roles of LLMs and what are the final components for solving graph-related problems, we classify LLM on graph techniques into three main categories:

LLM as Predictor: This category of methods serves LLM as the final component to output representations or predictions. It can be enhanced with GNNs and can be classified depending on how the graph information is injected into LLM: 1) *Graph as Sequence:* This type of method makes no changes to the LLM architecture, but makes it be aware of graph structure by taking a "graph token sequence" as input. The "graph token sequence" can be natural language descriptions for a graph or hidden representations outputted by graph encoders. 2) *Graph-Empowered LLM:* This type of method modifies the architecture of the LLM base model (i.e., Transformers) and enables it to conduct joint text and graph encoding inside their architecture. 3) *Graph-Aware LLM Finetuning:* This type of method makes no changes to the input of the LLMs or LLM architectures, but only fine-tunes the LLMs with supervision from the graph.

LLM as Encoder: This method is mostly utilized for graphs where nodes or edges are associated with text information (solving node-level or edge-level tasks). GNNs are the final components and we adopt LLM as the initial text encoder. To be specific, LLMs are first utilized to encode the text associated with the nodes/edges. The outputted feature vectors by LLMs then serve as input embeddings for GNNs for graph structure encoding. The output embeddings from the GNNs are adopted as final node/edge representations for downstream tasks. However, these methods suffer from convergence issues, sparse data issues, and inefficient issues, where we summarize solutions from optimization, data augmentation, and knowledge distillation perspectives.

LLM as Aligner: This category of methods adopts LLMs as text-encoding components and aligns them with GNNs which serve as graph structure encoding components. LLMs and GNNs are adopted together as the final components for task solving. To be specific, the alignment between LLMs and GNNs can be categorized into 1) *Prediction Alignment* where the generated pseudo labels from one modality are utilized for training on the other modality in an iterative learning fashion and 2) *Latent Space Alignment* where contrastive learning is adopted to align text embeddings generated by LLMs and graph embeddings generated by GNNs.

In the following sections, we will follow our categorization in Section III and discuss detailed methodologies for each graph scenario.

IV. Pure Graphs

The study of pure graphs in graph theory is essential for understanding the introduction of LLMs into graph-related reasoning problems. Pure graphs are a universal representation format used to address a wide range of algorithmic problems in computer science. Many graph-based concepts, such as shortest paths, specific sub-graphs, and flow networks, are strongly connected to real-world applications [132], [133], [134], [192]. Therefore, reasoning based on pure graphs is crucial for providing theoretical solutions and insights for real-world applications.

Nevertheless, many reasoning tasks require a computation capacity beyond traditional GNNs. GNNs are typically designed to carry out a bounded number of operations given a graph size. In contrast, graph reasoning problems can require up to indefinite complexity depending on the task's nature. On the other hand, LLMs demonstrate excellent emergent reasoning ability [47], [111], [112] recently. This is partially due to their autoregressive mechanism, which enables computing indefinite sequences of intermediate steps with careful prompting or training [47], [48].

The following subsections discuss the attempts to incorporate LLMs into pure graph reasoning problems. We will also discuss the corresponding challenges, limitations, and findings. Table 4 in the Appendix, available online lists a categorization of these efforts. Usually, input graphs are serialized as part of the input sequence, either by verbalizing the graph structure [123], [124], [125], [127], [128], [129], [130], [131] or by encoding the graph structure into implicit feature sequences [41]. The studied reasoning problems range from simpler ones like connectivity, shortest paths, and cycle detection to harder ones like maximum flow and Hamiltonian pathfinding (an NP-complete problem). A comprehensive list of the studied problems is listed in Appendix Table 5, available online. Note that we only list representative problems here. This table does not include more domain-specific problems, such as the spatial-temporal reasoning problems in [127]. We first briefly describe the approaches to formatting the graph inputs to be fed to LLMs.

Plainly Verbalizing Graphs: Verbalizing the graph structure in natural language is the most straightforward way of representing graphs. Representative approaches include describing the edge

and adjacency lists, widely studied in [123], [124], [127], [130]. For example, for a triangle graph with three nodes, the edge list can be written as "*[(0, 1), (1, 2), (2, 0)]*", which means node 0 is connected to node 1, node 1 is connected to node 2, node 2 is connected to node 0. It can also be written in natural language such as "*There is an edge between node 0 and node 1, an edge between node 1 and node 2, and an edge between node 2 and node 0.*" On the other hand, we can describe the adjacency list from the nodes' perspective. For example, for the same triangle graph, the adjacency list can be written as "*Node 0 is connected to node 1 and node 2. Node 1 is connected to node 0 and node 2. Node 2 is connected to node 0 and node 1.*"

Paraphrasing Graphs: The verbalized graphs can be lengthy, unstructured, and complicated to read, even for humans, so they might not be the best input format for LLMs to infer the answers. To this end, researchers also attempt to paraphrase the graph structure into more natural or concise sentences. [125] find that by prompting LLMs to generate a format explanation of the raw graph inputs for itself (*Format-Explanation*) or to pretend to play a role in a natural task (*Role Prompting*), the performance on some problems can be improved but not systematically. [130] explores the effect of grounding the pure graph in a real-world scenario, such as social networks, friendship graphs, or co-authorship graphs. In such graphs, nodes are described as people, and edges are relationships between people.

Encoding Graphs Into Implicit Feature Sequences: Finally, researchers also attempt to encode the graph structure into implicit feature sequences as part of the input sequence [41]. Unlike the previous verbalizing approaches, this usually involves training a graph encoder to encode the graph structure into a sequence of features and fine-tuning the LLMs to adapt to the new input format.

A. Direct Answering

Although graph-based reasoning problems usually involve complex computation, researchers still attempt to let language models directly generate answers from the serialized input graphs as a starting point, partially because of the simplicity of the approach and partially in awe of other emergent abilities of LLMs. Although various attempts have been made to optimize how graphs are presented in the input sequence discussed in the sections above, bounded by the finite sequence length and computational operations, this approach has a fundamental limitation to solving complex reasoning problems such as NP-complete ones. Unsurprisingly, most studies find that LLMs possess preliminary graph understanding ability, but the performance is less satisfactory on more complex problems or larger graphs [41], [123], [124], [125], [127], [130] where reasoning is necessary.

On plainly verbalized graphs, one can prompt LLMs to answer questions either in zero-shot or few-shot (in-context learning) settings. The former asks questions directly given the graph structure, while the latter asks questions about the graph structure after providing a few examples of questions and answers. [123], [124], [125] do confirm that LLMs can answer easier questions such as connectivity, neighbor identification, and graph size counting but fail to answer more complex questions such as cycle detection and Hamiltonian pathfinding. Their results also reveal that providing more examples in the few-shot setting increases the performance, especially on easier problems, although it is still not satisfactory. Results on paraphrased graphs indicate that encoding in real-world scenarios can improve performance on some problems, but it still cannot be done consistently. By encoding graphs into features, [41] demonstrates drastic performance improvement on problems including substructure counting, maximum triplet sum, shortest path, and bipartite matching. This indicates that fine-tuning LLMs has great fitting power on a specific task distribution.

B. Heuristic Reasoning

Direct mapping to the output leverages the LLMs' powerful representation power to "guess" the answers. Still, it does not fully utilize the LLMs' impressive emergent reasoning ability, which is essential for solving complex reasoning problems. To this end, attempts have been made to let LLMs perform heuristic reasoning on graphs. This approach encourages LLMs to perform a series of intermediate reasoning steps that might heuristically lead to the correct answer, which resembles a path-finding reasoning schema [202].

Reasoning Step by Step: Encouraged by the success of chain-of-thought (CoT) reasoning [47], [112], researchers also attempt to let LLMs perform reasoning step by step on graphs. Chain-of-thought encourages LLMs to roll out a sequence of reasoning steps to solve a problem, similar to how humans solve problems. Zero-shot CoT is a similar approach that does not require any examples. These techniques are studied in [41], [123], [124], [125], [127], [130], [131]. Results indicate that CoT-style reasoning can improve the performance on simpler problems, such as cycle detection and shortest path detection. Still, the improvement is inconsistent or diminishes on more complex problems, such as Hamiltonian path finding and topological sorting.

Retrieving Subgraphs as Evidence: Many graph reasoning problems, such as node degree counting and neighborhood detection, only involve reasoning on a subgraph of the whole graph. Such properties allow researchers to let LLMs retrieve the subgraphs as evidence and perform reasoning on the subgraphs. Build-a-Graph prompting [123] encourages LLMs to reconstruct the relevant graph structures and then perform reasoning on them. This method demonstrates promising results on problems except for Hamiltonian pathfinding, a notoriously tricky problem requiring reasoning on the whole graph. Another approach, Context-Summarization [125], encourages LLMs to summarize the key nodes, edges, or sub-graphs and perform reasoning.

Searching on Graphs: This kind of reasoning is related to the search algorithms on graphs, such as breadth-first search (BFS) and depth-first search (DFS) Although not universally applicable, BFS and DFS are the most intuitive and effective ways to solve some graph reasoning problems. Numerous explorations have been made to simulate searching-based reasoning, especially on knowledge-graph question answering. This approach enjoys the advantage of providing interpretable evidence besides the answer. Reasoning-on-Graphs (RoG) [128]

is a representative approach that prompts LLMs to generate several relation paths as plans, which are then retrieved from the knowledge graph (KG) and used as evidence to answer the questions. Another approach is to iteratively retrieve and reason on the subgraphs from KG [129], [131], simulating a dynamic searching process. At each step, the LLMs retrieve neighbors of the current nodes and then decide to answer the question or continue the next search step. These methods address the scalability challenge when knowledge from multiple graphs is available.

C. Algorithmic Reasoning

The previous two approaches are heuristic, which means that the reasoning process accords with human intuition but is not guaranteed to lead to the correct answer. In contrast, these problems are usually solved by algorithms in computer science. Therefore, researchers also attempt to let LLMs perform algorithmic reasoning on graphs. [123] proposed "*Algorithmic Prompting*", which prompts the LLMs to recall the algorithms that are relevant to the questions and then perform reasoning step by step according to the algorithms. Their results, however, do not show consistent improvement over the heuristic reasoning approach. A more direct approach, Graph-ToolFormer [126], lets LLMs generate API calls as explicit reasoning steps. These API calls are then executed externally to acquire answers on an external graph. This approach is suitable for converting real-world tasks into pure graph reasoning problems, and it has demonstrated efficacy in various applications such as knowledge graphs, social networks, and recommendation systems.

D. Discussion

Despite the extensive research, there has not been a consensus about the best practice in graph representation in LLMs. The eventual solution to this problem should reach a perfect balance between computation efficiency and information completeness, probably drawing inspiration from long-context LLM researches [209], [210]. The above reasoning methods are not mutually exclusive, and future efforts can be made to combine them to achieve better performance. For example, efficiency in algorithmic searching can be improved by prompting language models for better heuristics.

V. Text-Attributed Graphs

Text-attributed graphs exist ubiquitously in the real world, e.g., academic networks, and legal case networks. Learning on such networks requires the model to encode both the textual information associated with the nodes/edges and the structure information lying inside the input graph. Depending on the role of LLM, existing works can be categorized into three types: LLM as Predictor, LLM as Encoder, and LLM as Aligner. We summarize all surveyed methods in Appendix Table 6, available online.

A. LLM as Predictor

These methods serve the language model as the main model architecture to capture both the text information and graph structure information. They can be categorized into three types: *Graph as Sequence methods*, *Graph-Empowered LLMs*, and *Graph-Aware LLM finetuning methods*, depending on how structure information in graphs is injected into language models (input versus architecture versus loss). In the *Graph as Sequence methods*, graphs are converted into sequences that can be understood by language models together with texts from the inputs. In the *Graph-Empowered LLMs* methods, people modify the architecture of Transformers (which is the base architecture for LLMs) to enable it to encode text and graph structure simultaneously. In the *Graph-Aware LLM finetuning methods*, LLM is fine-tuned with graph structure supervision and can generate graph-contextualized representations.

1) Graph as Sequence: In these methods, the graph information is mainly encoded into the LLM from the "input" side. The ego-graphs associated with nodes/edges are serialized into a sequence $\mathbf{H}_{\mathcal{G}_v}$ which can be fed into the LLM together with the texts d_v:

$$\mathbf{H}_{\mathcal{G}_v} = \text{Graph2Seq}(\mathcal{G}_v), \tag{3}$$

$$\mathbf{h}_v = \text{LLM}([\mathbf{H}_{\mathcal{G}_v}, d_v]). \tag{4}$$

Depending on the choice of Graph2Seq$(\cdot)$ function, the methods can be further categorized into rule-based methods and GNN-based methods. The illustration of the categories can be found in Fig. 3.

Rule-Based: Linearizing Graphs into Text Sequence with Rules: These methods design rules to describe the structure with natural language and adopt a text prompt template as Graph2Seq$(\cdot)$. For example, given an ego-graph $\mathcal{G}_{v_i}$ of the paper node v_i connecting to author nodes v_j and v_k and venue nodes v_t and v_s, $\mathbf{H}_{\mathcal{G}_{v_i}} = \text{Graph2Seq}(\mathcal{G}_{v_i})$ = "*The centor paper node is v_i. Its author neighbor nodes are v_j and v_k and its venue neighbor nodes are v_t and v_s*". This is the most straightforward and easiest way (without introducing extra model parameters) to encode graph structures into language models. Along this line, InstructGLM [45] designs templates to describe local ego-graph structure (maximum 3-hop connection) for each node and conduct instruction tuning for node classification and link prediction. GraphText [64] further proposes a syntax tree-based method to transfer structure into text sequence. Researchers [81] also study when and why the linearized structure information on graphs can improve the performance of LLM on node classification and find that the structure information is beneficial when the textual information associated with the node is scarce (in this case, the structure information can provide auxiliary information gain).

GNN-Based: Encoding Graphs into Special Tokens with GNNs: Different from rule-based methods which use natural language prompts to linearize graphs into sequences, GNN-based methods adopt graph encoder models (i.e., GNN) to encode the ego-graph associated with nodes into special token representations which are concatenated with the pure text information into

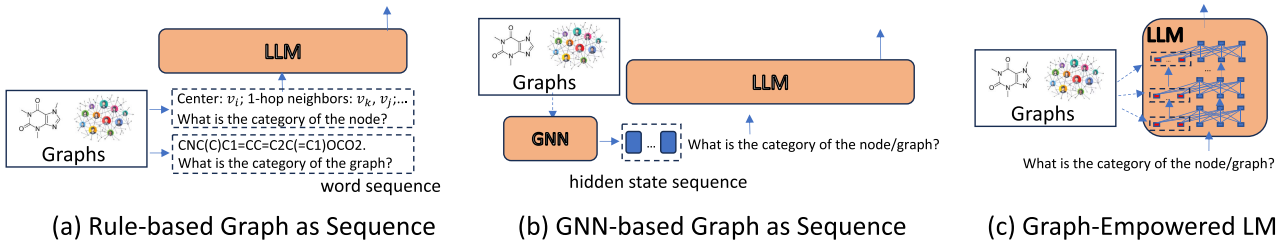


Fig. 3. The illustration of various LLM as Predictor methods, including (a) Rule-based Graph As Sequence, (b) GNN-based Graph As Sequence, (c) Graph-Empowered LLMs.

the language model:

$$\mathbf{H}_{\mathcal{G}_v} = \text{Graph2Seq}(\mathcal{G}_v) = \text{GraphEnc}(\mathcal{G}_v). \tag{5}$$

The strength of these methods is they can capture hidden representations of useful structure information with a strong graph encoder, while the challenge is how to fill the gap between graph modality and text modality. GNP [40] adopts a similar philosophy from LLaVA [90], where they utilize GNN to generate graph tokens and then project the graph tokens into the text token space with learnable projection matrices. The projected graph tokens are concatenated with text tokens and fed into the language model. GraphGPT [44] further proposes to train a text-grounded GNN for the projection with a text encoder and contrastive learning. DGTL [75] introduces disentangled graph learning, serves graph representations as positional encoding, and adds them to the text sequence. METERN [74] adds learnable relation embeddings to node textual sequences for text-based multiplex representation learning on graphs [91].

2) Graph-Empowered LLMs: In these methods, researchers design advanced LLM architecture (i.e., Graph-Empowered LLMs) which can conduct joint text and graph encoding inside their model architecture. Transformers [92] serve as the base model for nowadays pretrained LMs [23] and LLMs [35]. However, they are designed for natural language (sequence) encoding and do not take non-sequential structure information into consideration. To this end, Graph-Empowered LLMs are proposed. They have a shared philosophy of introducing virtual structure tokens $\boldsymbol{H}_{\mathcal{G}_v}$ inside each Transformer layer:

$$\widetilde{\boldsymbol{H}}_{d_v}^{(l)} = \left[\boldsymbol{H}_{\mathcal{G}_v}^{(l)}, \boldsymbol{H}_{d_v}^{(l)}\right] \tag{6}$$

where $\boldsymbol{H}_{\mathcal{G}_v}$ can be learnable embeddings or output from graph encoders. Then the original multi-head attention (MHA) in Transformers is modified into an asymmetric MHA to take the structure tokens into consideration:

$$\text{MHA}_{asy}\left(\boldsymbol{H}_{d_v}^{(l)}, \widetilde{\boldsymbol{H}}_{d_v}^{(l)}\right) = \|_{u=1}^{U} \text{head}_u\left(\boldsymbol{H}_{d_v}^{(l)}, \widetilde{\boldsymbol{H}}_{d_v}^{(l)}\right),$$

$$\text{where head}_u\left(\boldsymbol{H}_{d_v}^{(l)}, \widetilde{\boldsymbol{H}}_{d_v}^{(l)}\right) = \text{softmax}\left(\frac{\boldsymbol{Q}_u^{(l)}\widetilde{\boldsymbol{K}}_u^{(l)\top}}{\sqrt{d/U}}\right)\cdot \widetilde{\boldsymbol{V}}_u^{(l)},$$

$$\boldsymbol{Q}_u^{(l)} = \boldsymbol{H}_{d_v}^{(l)}\boldsymbol{W}_{Q,u}^{(l)},\quad \widetilde{\boldsymbol{K}}_u^{(l)} = \widetilde{\boldsymbol{H}}_{d_v}^{(l)}\boldsymbol{W}_{K,u}^{(l)},\quad \widetilde{\boldsymbol{V}}_u^{(l)} = \widetilde{\boldsymbol{H}}_{d_v}^{(l)}\boldsymbol{W}_{V,u}^{(l)}. \tag{7}$$

With the asymmetric MHA mechanism, the node encoding process of the $(l+1)$-th layer will be:

$$\widetilde{\boldsymbol{H}}_{d_v}^{(l)'} = \text{Normalize}\left(\boldsymbol{H}_{d_v}^{(l)} + \text{MHA}_{asy}\left(\widetilde{\boldsymbol{H}}_{d_v}^{(l)}, \boldsymbol{H}_{d_v}^{(l)}\right)\right),$$

$$\boldsymbol{H}_{d_v}^{(l+1)} = \text{Normalize}\left(\widetilde{\boldsymbol{H}}_{d_v}^{(l)'} + \text{MLP}\left(\widetilde{\boldsymbol{H}}_{d_v}^{(l)'}\right)\right). \tag{8}$$

Along this line of work, GreaseLM [66] proposes to have a language encoding component and a graph encoding component in each layer. These two components interact through a modality-fusion layer (MInt layer), where a special structure token is added to the text Transformer input, and a special node is added to the graph encoding layer. DRAGON [80] further proposes strategies to pretrain GreaseLM with unsupervised signals. GraphFormers [71] are designed for node representation learning on homogeneous text-attributed networks where the current layer `[CLS]` token hidden states of neighboring documents are aggregated and added as a new token on the current layer center node text encoding. Patton [31] proposes to pretrain GraphFormers with two novel strategies: network-contextualized masked language modeling and masked node prediction. Heterformer [72] introduces virtual neighbor tokens for text-rich neighbors and textless neighbors which are concatenated with the original text tokens and fed into each Transformer layer. Edgeformers [73] are proposed for representation learning on textual-edge networks where edges are associated with rich textual information. When conducting edge encoding, virtual node tokens will be concatenated onto the original edge text tokens for joint encoding.

3) Graph-Aware LLM Finetuning: In these methods, the graph information is mainly injected into the LLM by "finetuning on graphs". Researchers assume that the structure of graphs can provide hints on what documents are "semantically similar" to what other documents. For example, papers citing each other in an academic graph can be of similar topics. These methods adopt vanilla language models that take text as input (e.g., BERT [23] and SciBERT [25]) as the base model and finetune them with structure signals on the graph [50]. After that, the LLMs will learn node/edge representations that capture the graph homophily from the text perspective. This is the simplest way to utilize LLMs on graphs. However, during encoding, the model itself can only consider text.

Most methods adopt the two-tower encoding and training pipeline, where the representation of each node is obtained

separately and the model is optimized as follows:

$$\mathbf{h}_{v_i} = \text{LLM}_\theta(d_{v_i}), \quad \min_\theta f(\mathbf{h}_{v_i}, \{\mathbf{h}_{v_i^+}\}, \{\mathbf{h}_{v_i^-}\}). \tag{9}$$

Here v_i^+ represents the positive nodes to v_i, v_i^- represents the negative nodes to v_i and $f(\cdot)$ denotes the pairwise training objective. Different methods have different strategies for v_i^+ and v_i^- with different training objectives $f(\cdot)$. SPECTER [50] constructs the positive text/node pairs with the citation relation, explores random negatives and structure hard negatives, and fine-tunes SciBERT [25] with the triplet loss. SciNCL [51] extends SPECTER by introducing more advanced positive and negative sampling methods based on embeddings trained on graphs. Touchup-G [53] proposes the measurement of feature homophily on graphs and brings up a binary cross-entropy fine-tuning objective. TwHIN-BERT [55] mines positive node pairs with off-the-shelf heterogeneous information network embeddings and trains the model with a contrastive social loss. MICoL [58] discovers semantically positive node pairs with meta-path [89] and adopts the InfoNCE objective. E2EG [59] utilizes a similar philosophy from GIANT [57] and adds a neighbor prediction objective apart from the downstream task objective. WalkLM [60] conducts random walks for structure linearization before fine-tuning the language model. A summarization of the two-tower graph-centric LLM fine-tuning objectives can be found in Appendix Table 7, available online.

There are other methods using the one-tower pipeline, where node pairs are concatenated and encoded together:

$$\mathbf{h}_{v_i,v_j} = \text{LLM}_\theta(d_{v_i}, d_{v_j}), \quad \min_\theta f(\mathbf{h}_{v_i,v_j}). \tag{10}$$

LinkBERT [30] proposes a document relation prediction objective (an extension of next sentence prediction in BERT [23]) which aims to classify the relation of two node text pairs from contiguous, random, and linked. MICoL [58] explores predicting the node pairs' binary meta-path or meta-graph indicated relation with the one-tower language model.

4) Discussion: Although the community is making good progress, there are still some open questions to be solved.

Graph as Code Sequence: Existing graphs as sequence methods are mainly rule-based or GNN-based. The former relies on natural language to describe the graphs which is not natural for structure data, while the latter has a GNN component that needs to be trained. A more promising way is to obtain a structure-aware sequence for graphs that can support zero-shot inference. A potential solution is to adopt codes (that can capture structures, e.g., graph XML or JSON) to describe the graphs and utilize code LLMs [22].

Advanced Graph-Empowered LLM Techniques: Graph-empowered LLM is a promising direction to achieve foundational models for graphs. However, existing works are far from enough: 1) *Task*. Existing methods are mainly designed for representation learning (with encoder-only LLMs) which are hard to adopt for generation tasks. A potential solution is to design Graph-Empowered LLMs with decoder-only or encoder-decoder LLMs as the base architecture. 2) *Pretraining*. Pretraining is important to enable LLMs with contextualized data understanding capability, which can be generalized to other tasks. However, existing works mainly focus on pretraining LLMs on homogeneous text-attributed networks. Future studies are needed to explore LLM pretraining in more diverse real-world scenarios including heterogeneous text-attributed networks [72], dynamic text-attributed networks [127], and textual-edge networks [73].

B. LLM as Encoder

LLMs extract textual features to serve as initial node feature vectors for GNNs, which then generate node/edge representations and make predictions. These methods typically adopt an LLM-GNN cascaded architecture to obtain the final representation $\mathbf{h}_{v_i}$ for node v_i:

$$\mathbf{x}_{v_i} = \text{LLM}(d_{v_i}) \quad \mathbf{h}_{v_i} = \text{GNN}(\mathbf{X}_v, \mathcal{G}). \tag{11}$$

Here $\mathbf{x}_{v_i}$ is the feature vector that captures the textual information d_{v_i} associated with v_i. The final representation $\mathbf{h}_{v_i}$ will contain both textual information and structure information of v_i and can be used for downstream tasks. In the following sections, we will discuss the optimization, augmentation, and distillation of such models. The figures for these techniques can be found in Fig. 4.

1) Optimization: One-Step Training refers to training the LLM and GNN together in the cascaded architecture for the downstream tasks. TextGNN [76] explores GCN [83], GraphSAGE [84], GAT [85] as the base GNN architecture, adds skip connection between LLM output and GNN output, and optimizes the whole architecture for sponsored search task. AdsGNN [77] further extends TextGNN by proposing edge-level information aggregation. GNN-LM [65] adds GNN layers to enable the vanilla language model to reference similar contexts in the corpus for language modeling. Joint training LLMs and GNNs in a cascaded pipeline is convenient but may suffer from efficiency [67] (only support sampling a few one-hop neighbors regarding memory complexity) and local minimal [34] (LLM underfits the data) issues.

Two-Step Training means first adapting LLMs to the graph, and then finetuning the whole LLM-GNN cascaded pipeline. GIANT [57] proposes to conduct neighborhood prediction with the use of XR-Transformers [78] and results in an LLM that can output better feature vectors than bag-of-words and vanilla BERT [23] embedding for node classification. LM-GNN [67] introduces graph-aware pre-fine-tuning to warm up the LLM on the given graph before fine-tuning the whole LLM-GNN pipeline and demonstrating significant performance gain. SimTeG [34] finds that the simple framework of first training the LLMs on the downstream task and then fixing the LLMs and training the GNNs can result in outstanding performance. They further find that using the efficient fine-tuning method, e.g., LoRA [39] to tune the LLM can alleviate overfitting issues. GaLM [79] explores ways to pretrain the LLM-GNN cascaded

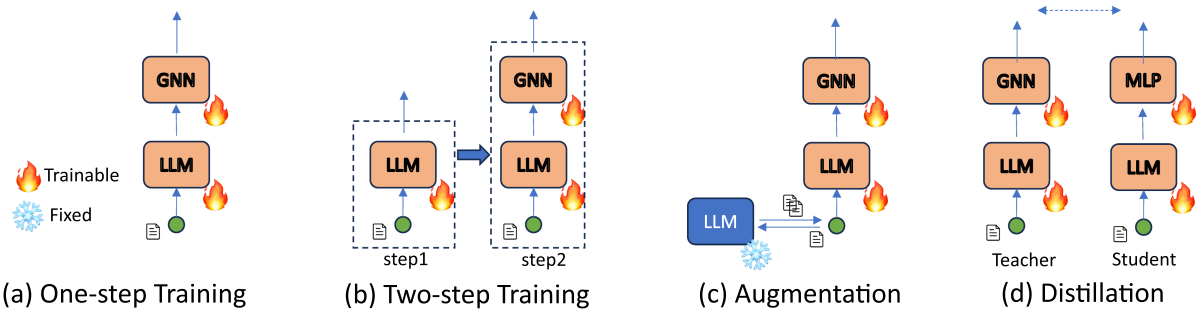


Fig. 4. The illustration of various techniques related to LLM as Encoder, including (a) One-step Training, (b) Two-step Training, (c) Data Augmentation, and (d) Knowledge Distillation.

architecture. The two-step strategy can effectively alleviate the insufficient training of the LLM which contributes to higher text representation quality but is more computationally expensive and time-consuming than the one-step training strategy.

2) Data Augmentation: With its demonstrated zero-shot capability [42], LLMs can be used for data augmentation to generate additional text data for the LLM-GNN cascaded architecture. The philosophy of using LLM to generate pseudo data is widely explored in NLP [82], [88]. LLM-GNN [63] proposes to conduct zero-shot node classification on text-attributed networks by labeling a few nodes and using the pseudo labels to fine-tune GNNs. TAPE [69] presents a method that uses LLM to generate prediction text and explanation text, which serve as augmented text data compared with the original text data. A following medium-scale language model is adopted to encode the texts and output features for augmented texts and original text respectively before feeding into GNNs. ENG [70] brings forward the idea of generating labeled nodes for each category, adding edges between labeled nodes and other nodes, and conducting semi-supervised GNN learning for node classification.

3) Knowledge Distillation: LLM-GNN cascaded pipeline is capable of capturing both text information and structure information. However, the pipeline suffers from time complexity issues during inference, since GNNs need to conduct neighbor sampling and LLMs need to encode the text associated with both the center node and its neighbors. A straightforward solution is to serve the LLM-GNN cascade pipeline as the teacher model and distill it into an LLM as the student model. In this case, during inference, the model (which is a pure LLM) only needs to encode the text on the center node and avoid time-consuming neighbor sampling. AdsGNN [77] proposes an L2-loss to force the outputs of the student model to preserve topology after the teacher model is trained. GraD [68] introduces three strategies including the distillation objective and task objective to optimize the teacher model and distill its capability to the student model.

4) Discussion: Given that GNNs are demonstrated as powerful models in encoding graphs, "LLMs as encoders" seems to be the most straightforward way to utilize LLMs on graphs. However, there are still open questions.

Limited Task: Go Beyond Representation Learning: Current "LLMs as encoders" methods or LLM-GNN cascaded architectures are mainly focusing on representation learning, given the single embedding propagation-aggregation mechanism of GNNs, which prevents it from being adopted to generation tasks (e.g., node/text generation). A potential solution to this challenge can be to conduct GNN encoding for LLM-generated token-level representations and to design proper decoders that can perform generation based on the LLM-GNN cascaded model outputs.

Low Efficiency: Advanced Knowledge Distillation: The LLM-GNN cascaded pipeline suffers from time complexity issues since the model needs to conduct neighbor sampling and then embedding encoding for each neighboring node. Although there are methods that explore distilling the learned LLM-GNN model into an LLM model for fast inference, they are far from enough given that the inference of LLM itself is time-consuming. A potential solution is to distill the model into a much smaller LM or even an MLP. Similar methods [86] have been proven effective in GNN to MLP distillation and are worth exploring for the LLM-GNN cascaded pipeline as well.

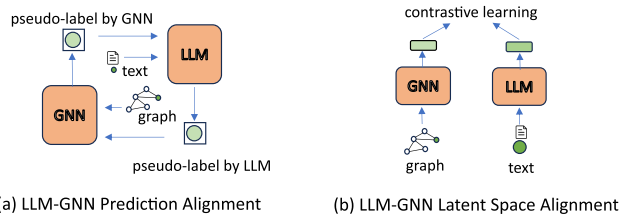


Fig. 5. The illustration of LLM as Aligner methods, including (a) LLM-GNN Prediction Alignment and (b) LLM-GNN Latent Space Alignment.

C. LLM as Aligner

These methods contain an LLM component for text encoding and a GNN component for structure encoding. These two components are served equally and trained iteratively or parallelly. LLMs and GNNs can mutually enhance each other since the LLMs can provide textual signals to GNNs, while the GNNs can deliver structure information to LLMs. According to how the LLM and the GNN interact, these methods can be further categorized into: LLM-GNN Prediction Alignment and LLM-GNN Latent Space Alignment. The illustration of these two categories of methods can be found in Fig. 5.

1) LLM-GNN Prediction Alignment: This refers to training the LLM with the text data on a graph and training the GNN

with the structure data on a graph iteratively. LLM will generate labels for nodes from the text perspective and serve them as pseudo-labels for GNN training, while GNN will generate labels for nodes from the structure perspective and serve them as pseudo-labels for LLM training. By this design, these two modality encoders can learn from each other and contribute to a final joint text and graph encoding. In this direction, LTRN [56] proposes a novel GNN architecture with personalized PageRank [93] and attention mechanism for structure encoding while adopting BERT [23] as the language model. The pseudo labels generated by LLM and GNN are merged for the next iteration of training. GLEM [61] formulates the iterative training process into a pseudo-likelihood variational framework, where the E-step is to optimize LLM and the M-step is to train the GNN.

2) LLM-GNN Latent Space Alignment: It denotes connecting text encoding (LLM) and structure encoding (GNN) with cross-modality contrastive learning:

$$\mathbf{h}_{d_{v_i}} = \text{LLM}(d_{v_i}), \mathbf{h}_{v_i} = \text{GNN}(\mathcal{G}_v), \tag{12}$$

$$l(\mathbf{h}_{d_{v_i}}, \mathbf{h}_{v_i}) = \frac{\text{Sim}(\mathbf{h}_{d_{v_i}}, \mathbf{h}_{v_i})}{\sum_{j \neq i} \text{Sim}(\mathbf{h}_{d_{v_i}}, \mathbf{h}_{v_j})}, \tag{13}$$

$$\mathcal{L} = \sum_{v_i \in \mathcal{G}} \frac{1}{2|\mathcal{G}|} (l(\mathbf{h}_{d_{v_i}}, \mathbf{h}_{v_i}) + l(\mathbf{h}_{v_i}, \mathbf{h}_{d_{v_i}})) \tag{14}$$

A similar philosophy is widely used in vision-language joint modality learning [95]. Along this line of approaches, ConGrat [52] adopts GAT [85] as the graph encoder and tries MPNet [33] as the language model encoder. They have expanded the original InfoNCE loss by incorporating graph-specific elements. These elements pertain to the most likely second, third, and subsequent choices regarding the nodes from which a text originates and the texts that a node generates. In addition to the node-level multi-modality contrastive objective, GRENADE [54] proposes KL-divergence-based neighbor-level knowledge alignment: minimize the neighborhood similarity distribution calculated between LLM and GNN. G2P2 [62] further extends node-text contrastive learning by adding text-summary interaction and node-summary interaction. Then, they introduce using label texts in the text modality for zero-shot classification, and using soft prompts for few-show classification. THLM [32] proposes to pretrain the language model by contrastive learning with a heterogeneous GNN on heterogeneous text-attributed networks. The pretrained LLM can be fine-tuned on downstream tasks.

3) Discussion: Most existing methods adopt homogeneous text-graph alignment, assuming that the semantic relation between the two modalities, namely text and graph, is singular. However, this is not usually the case in the real world, given: 1) The existence of multimodal attributes: Other modalities, e.g., images can appear together with text and graph. In this case, it is worth researching how to align the multimodal attributes in a graph scenario. 2) Heterogeneous semantic relations: the semantic relationships between data units (text/image/graph) can be multiplex. Different relations have different distributions and a single semantic alignment will fail to capture the comprehensively [74].

VI. Text-Paired Graphs

Graphs are prevalent data objects in scientific disciplines such as cheminformatics [182], [193], [199], material informatics [180], bioinformatics [200], and computer vision [146]. Within these diverse fields, graphs frequently come paired with critical graph-level text information. For instance, molecular graphs in cheminformatics are annotated with text properties such as toxicity, water solubility, and permeability properties [180], [182]. Research on such graphs (scientific discovery) could be accelerated by the text information and the adoption of LLMs. In this section, we review the application of LLMs on graph-captioned graphs with a focus on molecular graphs. According to the technique categorization in Section III-B, we begin by investigating methods that utilize LLMs as Predictor. Then, we discuss methods that align GNNs with LLMs. We summarize all surveyed methods in Appendix Table 8 and Figure 6, available online.

A. LLM as Predictor

In this subsection, we review how to conduct "LLM as Predictor" for graph-level tasks. Existing methods can be categorized into Graph as Sequence (treat graph data as sequence input) and Graph-Empowered LLMs (design model architecture to encode graphs).

1) Graph as Sequence: For text-paired graphs, we have three steps to utilize existing LLM for graph inputs. **Step 1**: Linearize graphs into sequence with rule-based methods. **Step 2**: Tokenize the linearized sequence. **Step 3**: Train/Finetune different LLMs (e.g., Encoder-only, Encoder-Decoder, Decoder-only) for specific tasks. We will discuss each step as follows.

Step 1: Rule-based Graph Linearization. Rule-based linearization converts molecular graphs into text sequences that can be processed by LLMs. To achieve this, researchers develop specifications based on human expertise in the form of line notations [147]. For example, the Simplified Molecular-Input Line-Entry System (SMILES) [147] records the symbols of nodes encountered during a depth-first traversal of a molecular graph. The International Chemical Identifier (InChI) [148] encodes molecular structures into unique string texts with more hierarchical information. Canonicalization algorithms produce unique SMILES for each molecule, often referred to as canonical SMILES. However, there are more than one SMILES corresponding to a single molecule and SMILES sometimes represent invalid molecules; LLMs learned from these linearized sequences can easily generate invalid molecules (e.g., incorrect ring closure symbols and unmatched parentheses) due to syntactical errors. To this end, DeepSMILES [149] is proposed. It can alleviate this issue in most cases but does not guarantee 100% robustness. The linearized string could still violate basic physical constraints. To fully address this problem, SELFIES [150] is introduced which consistently yields valid molecular graphs.

Step 2: Tokenization. These approaches for linearized sequences are typically language-independent. They operate at both character level [166], [177] and substring level [161], [168], [172], [173], [174], [175], based on SentencePiece or BPE [154].

Additionally, RT [163] proposes a tokenization approach that facilitates handling regression tasks within LM Transformers.

Step 3: Encoding the Linearized Graph with LLMs.

Encoder-only LLMs: Earlier LLMs like SciBERT [25] and BioBERT [179] are trained on scientific literature to understand natural language descriptions related to molecules but are not capable of comprehending molecular graph structures. To this end, SMILES-BERT [178] and MFBERT [175] are proposed for molecular graph classification with linearized SMILES strings. Since scientific natural language descriptions contain human expertise which can serve as a supplement for molecular graph structures, recent advances emphasize joint understanding of them [158], [174]: The linearized graph sequence is concatenated with the raw natural language data and then input into the LLMs. Specifically, KV-PLM [174] is built based on BERT [23] to understand the molecular structure in a biomedical context. CatBERTa [158], as developed from RoBERTa [24], specializes in the prediction of catalyst properties for molecular graphs.

Encoder-Decoder LLMs: Encoder-only LLMs may lack the capability for generation tasks. In this section, we discuss LLMs with encoder-decoder architectures. For example, Chemformer [155] uses a similar architecture as BART [28]. The representation from the encoder can be used for property prediction tasks, and the whole encoder-decoder architecture can be optimized for molecule generation. Others focus on molecule captioning (which involves generating textual descriptions from a molecule) and text-based molecular generation (where a molecular graph structure is generated from a natural description). Specifically, MolT5 [122] is developed based on the T5 [29], suitable for these two tasks. It formulates molecule-text translation as a multilingual problem and initializes the model using the T5 checkpoint. The model was pre-trained on two monolingual corpora: the Colossal Clean Crawled Corpus (C4) [29] for the natural language modality and one million SMILES [155] for the molecule modality. Text+Chem T5 [170] extends the input and output domains to include both SMILES and texts, unlocking LLMs for more generation functions such as text or reaction generation. ChatMol [165] exploits the interactive capabilities of LLMs and proposes designing molecule structures through multi-turn dialogs with T5.

Decoder-only LLMs: Decoder-only architectures have been adopted for recent LLMs due to their advanced generation ability. MolGPT [176] and MolXPT [168] are GPT-style models used for molecule classification and generation. Specifically, MolGPT [176] focuses on conditional molecule generation tasks using scaffolds, while MolXPT [168] formulates the classification task as a question-answering problem with yes or no responses. RT [163] adopts XLNet [27] and focuses on molecular regression tasks. It frames the regression as a conditional sequence modeling problem. Galactica [177] is a set of LLMs with a maximum of 120 billion parameters, which is pretrained on two million compounds from PubChem [182]. Therefore, Galactica could understand molecular graph structures through SMILES. With instruction tuning data and domain knowledge, researchers also adapt general-domain LLMs such as LLaMA to recognize molecular graph structures and solve molecule tasks [159]. Recent studies also explore the in-context learning capabilities of LLMs on graphs. LLM-ICL [167] assesses the performance of LLMs across eight tasks in the molecular domain, ranging from property classification to molecule-text translation. MolReGPT [164] proposes a method to retrieve molecules with similar structures and descriptions to improve in-context learning. LLM4Mol [162] utilizes the summarization capability of LLMs as a feature extractor and combines it with a smaller, tunable LLM for specific prediction tasks.

2) Graph-Empowered LLMs: Different from the methods that adopt the original LLM architecture (i.e., Transformers) and input the graphs as sequences to LLMs, graph-empowered LLMs attempt to design LLM architectures that can conduct joint encoding of text and graph structures. Some works modify the positional encoding of Transformers. For instance, GIMLET [46] treats nodes in a graph as tokens. It uses one Transformer to manage both the graph structure and text sequence $[v_1, v_2, \ldots, v_{|\mathcal{V}|}, s_{|\mathcal{V}|+1}, \ldots, s_{|\mathcal{V}|+|d_{\mathcal{G}}|}]$, where $v \in \mathcal{V}$ is a node and $s \in d_{\mathcal{G}}$ is a token in the text associated with $\mathcal{G}$. This sequence cannot reflect graph structure. Therefore, a new position encoding (PE) is used to jointly encode graph structures and text sequences. It defines the relative distance between tokens i and j as follows:

$$\mathrm{PE}(i,j) = \begin{cases} i-j & \text{if } i,j \in d_{\mathcal{G}}, \\ \mathrm{GSD}(i,j) + \mathrm{Mean}_{e_k \in \mathrm{SP}(i,j)} \mathbf{x}_{e_k} & \text{if } i,j \in \mathcal{V}, \\ -\infty & \text{if } i \in \mathcal{V}, j \in d_{\mathcal{G}}, \\ 0 & \text{if } i \in d_{\mathcal{G}}, j \in \mathcal{V}. \end{cases} \tag{15}$$

GSD is the graph shortest distance between two nodes, and $\mathrm{Mean}_{k \in \mathrm{SP}(i,j)}$ represents the mean pooling of the edge features $\mathbf{x}_{e_k}$ along the shortest path $\mathrm{SP}(i,j)$ between nodes i and j. GIMLET [46] adapts bi-directional attention for node tokens and enables texts to selectively attend to nodes. These designs render the Transformer's submodule, which handles the graph part, equivalent to a Graph Transformer [140].

Cross-attention is also used to interact representations between graphs and texts. Given the graph hidden state $\mathbf{h}_{\mathcal{G}}$, its node-level hidden state $\mathbf{H}_v$ and text hidden state $\mathbf{H}_{d_{\mathcal{G}}}$, Text2Mol [121] implemented interaction between representations in the hidden layers of encoders, while Prot2Text [160] implemented this interaction within the layers of between encoder and decoder $\mathbf{H}_{d_{\mathcal{G}}} = \mathrm{softmax}(\frac{\mathbf{W}_Q \mathbf{H}_{d_{\mathcal{G}}} \cdot (\mathbf{W}_K \mathbf{H}_v)^T}{\sqrt{d_k}}) \cdot \mathbf{W}_V \mathbf{H}_v$, where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$ are trainable parameters that transform the query modality (e.g., sequences) and the key/value modality (e.g., graphs) into the attention space. Furthermore, Prot2Text [160] utilizes two trainable parameter matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ to integrate the graph representation into the sequence representation $\mathbf{H}_{d_{\mathcal{G}}} = (\mathbf{H}_{d_{\mathcal{G}}} + \mathbf{1}_{|d_{\mathcal{G}}|} \mathbf{h}_{\mathcal{G}} \mathbf{W}_1) \mathbf{W}_2$.

3) Discussion: LLM Inputs with Sequence Prior: The first challenge is that the progress in advanced linearization methods has not progressed in tandem with the development of LLMs. Emerging around 2020, linearization methods for molecular graphs like SELFIES offer significant grammatical advantages, yet advanced LMs and LLMs from graph machine learning and language model communities might not fully utilize these, as these encoded results are not part of pretraining corpora prior to their proposal. Consequently, recent studies [167] indicate that

LLMs, such as GPT-3.5/4, may be less adept at using SELFIES compared to SMILES. Therefore, the performance of LM-only and LLM-only methods may be limited by the expressiveness of older linearization methods, as there is no way to optimize these hard-coded rules during the learning pipeline of LLMs. *However, the second challenge remains as the inductive bias of graphs may be broken by linearization*. Rule-based linearization methods introduce inductive biases for sequence modeling, thereby breaking the permutation invariance assumption inherent in molecular graphs. It may reduce task difficulty by introducing sequence order to reduce the search space. However, it does not mean model generalization. Specifically, there could be multiple string-based representations for a single graph from single or different approaches. Numerous studies [151], [152], [153] have shown that training on different string-based views of the same molecule can improve the sequential model's performance, as these data augmentation approaches manage to retain the permutation-invariance nature of graphs. These advantages are also achievable with a permutation-invariant GNN, potentially simplifying the model by reducing the need for complex, string-based data augmentation design.

LLM Inputs With Graph Prior: Rule-based linearization may be considered less expressive and generalizable compared to the direct graph representation with rich node features, edge features, and the adjacency matrix [186]. Various atomic features include atomic number, chirality, degree, formal charge, number of hydrogen atoms, number of radical electrons, hybridization state, aromaticity, and presence in a ring. Bond features encompass the bond's type (e.g., single, double, or triple), the bond's stereochemistry (e.g., E/Z or cis/trans), and whether the bond is conjugated [187]. Each feature provides specific information about atomic properties and structure, crucial for molecular modeling and cheminformatics. One may directly **vectorize** the molecular graph structure into binary vectors [185] and then apply parameterized Multilayer Perceptrons (MLPs) on the top of these vectors to get the graph representation. These vectorization approaches are based on human-defined rules and vary, such as MACCS, ECFP, and CDK fingerprints [185]. These rules take inputs of a molecule and output a vector consisting of 0/1 bits. Each bit denotes a specific type of substructure related to functional groups that could be used for various property predictions. Fingerprints consider atoms and structures, but they cannot automatically learn from the graph structure. GNNs could serve as automatic feature extractors to replace or enhance fingerprints. Some specific methods are explored in Section VI-A2, while the other graph prior such as the eigenvectors of a graph Laplacian and the random walk prior could also be used [141].

LLM Outputs for Prediction: LMs like KV-PLM [174], SMILES-BERT [178], MFBERT [175], and Chemformer [155] use a prediction head on the output vector of the last layer. These models are finetuned with standard classification and regression losses but may not fully utilize all the parameters and advantages of the complete architecture. In contrast, models like RT [163], MolXPT [168], and Text+Chem T5 [170] frame prediction as a text generation task. These models are trained with either masked language modeling or autoregressive targets, which requires a meticulous design of the context words in the text [163]. Specifically, domain knowledge instructions may be necessary to activate the in-context learning ability of LLMs, thereby making them domain experts [167]. For example, a possible template could be divided into four parts: {General Description}{Task-Specific Description}{Question-Answer Examples}{Test Question}.

LLM Outputs for Reasoning: Since string representations of molecular graphs usually carry new and in-depth domain knowledge, which is beyond the knowledge of LLMs, recent work [145], [156], [164] also attempts to utilize the reasoning ability of LLMs, instead of using them as a knowledge source for predicting the property of molecular graphs. ReLM [156] utilizes GNNs to suggest top-k candidates, which were then used to construct multiple-choice answers for in-context learning. ChemCrow [145] designs the LLMs as the chemical agent to implement various chemical tools. It avoided direct inference in an expertise-intensive domain.

B. LLM as Aligner

1) Latent Space Alignment: One may directly align the latent spaces of the GNN and LLM through contrastive learning and predictive regularization. Typically, a graph representation from a GNN can be read out by summarizing all node-level representations, and a sequence representation can be obtained from the [CLS] token. We first use two projection heads, which are usually MLPs, to map the separate representation vectors from the GNN and LLM into a unified space as $\mathbf{h}_{\mathcal{G}}$ and $\mathbf{h}_{d_{\mathcal{G}}}$, and then align them within this space. Specifically, MoMu [173] and MoMu-v2 [172] retrieve two sentences from the corpus for each molecular graph. During training, graph data augmentation was applied to molecular graphs, creating two augmented views. Consequently, there are four pairs of $\mathcal{G}$ and $d_{\mathcal{G}}$. For each pair, the contrastive loss for space alignment is as $\ell_{\text{MoMu}} = -\log \frac{\exp(\cos(\mathbf{h}_{\mathcal{G}}, \mathbf{h}_{d_{\mathcal{G}}})/\tau)}{\sum_{\tilde{d}_{\mathcal{G}} \neq d_{\mathcal{G}}} \exp(\cos(\mathbf{h}_{\mathcal{G}}, \mathbf{h}_{\tilde{d}_{\mathcal{G}}})/\tau)}$ where τ is the temperature hyper-parameter and $\tilde{d}_{\mathcal{G}}$ denotes the sequence not paired to the graph $\mathcal{G}$. MoleculeSTM [171] also applies contrastive learning to minimize the representation distance between a molecular graph $\mathcal{G}$ and its corresponding texts $d_{\mathcal{G}}$, while maximizing the distance between the molecule and unrelated descriptions. MoleculeSTM [171] randomly samples negative graphs or texts to construct negative pairs of $(\mathcal{G}, \tilde{d})$ and $(\tilde{\mathcal{G}}, d)$. Similarly, MolFM [161] and GIT-Mol [157] implement contrastive loss with mutual information and negative sampling. These two methods also use cross-entropy to regularize the unified space with the assumption that randomly permuted graph and text inputs are predictable if they originate from the same molecule.

However, the aforementioned methods cannot leverage task labels. Given a classification label y, CLAMP [169] learns to map active molecules $(y = 1)$ so that they align with the corresponding assay description for each molecular graph $\mathcal{G}$: $\ell_{\text{CLAMP}} = y \log(\sigma(\tau^{-1} \mathbf{h}_{\mathcal{G}}^T \mathbf{h}_{d_{\mathcal{G}}})) + (1 - y) \log(1 -$

TABLE II
DATA COLLECTION IN SECTION V FOR TEXT-ATTRIBUTED GRAPHS

Text.	Data	Year	Task	# Nodes	# Edges	Domain	Source & Notes
Node	ogb-arxiv	2020.5	NC	169,343	1,166,243	Academic	OGB [187]
	ogb-products	2020.5	NC	2,449,029	61,859,140	E-commerce	OGB [187]
	ogb-papers110M	2020.5	NC	111,059,956	1,615,685,872	Academic	OGB [187]
	ogb-citation2	2020.5	LP	2,927,963	30,561,187	Academic	OGB [187]
	Cora	2000	NC	2,708	5,429	Academic	[10]
	Citeseer	1998	NC	3,312	4,732	Academic	[11]
	DBLP	2023.1	NC, LP	5,259,858	36,630,661	Academic	www.aminer.org/citation
	MAG	2020	NC, LP, Rec RG	$\sim$ 10M	$\sim$ 50M	Academic	multiple domains [12] [13]
	Goodreads-books	2018	NC, LP	$\sim$ 2M	$\sim$ 20M	Books	multiple domains [14]
	Amazon-items	2018	NC, LP, Rec	$\sim$ 15.5M	$\sim$ 100M	E-commerce	multiple domains [15]
	SciDocs	2020	NC, UAP, LP, Rec	-	-	Academic	[50]
	PubMed	2020	NC	19,717	44,338	Academic	[16]
	Wikidata5M	2021	LP	$\sim$ 4M	$\sim$ 20M	Wikipedia	[17]
	Twitter	2023	NC, LP	176,279	2,373,956	Social	[52]
Edge	Goodreads-reviews	2018	EC, LP	$\sim$ 3M	$\sim$ 100M	Books	multiple domains [14]
	Amazon-reviews	2018	EC, LP	$\sim$ 15.5M	$\sim$ 200M	E-commerce	multiple domains [15]
	Stackoverflow	2023	EC, LP	129,322	281,657	Social	[73]

Task: "NC", "UAP", "LP", "Rec", "EC", "RG" denote node classification, user activity prediction, link prediction, recommendation, edge classification, and regression task.

$\sigma(\tau^{-1}\mathbf{h}_{\mathcal{G}}^T\mathbf{h}_{d_{\mathcal{G}}}))$. CLAMP [169] requires labels to encourage that active molecules and their corresponding text descriptions are clustered together in the latent space. To advance the alignment between two modalities, MolCA [166] trains the Query Transformer (Q-Former) [189] for molecule-text projecting and contrastive alignment. Q-former initializes N_q learnable query tokens $\{\mathbf{q}_k\}_{k=1}^{N_q}$. These query tokens are updated with self-attention and interact with the output of GNNs through cross-attention to obtain the k-th queried molecular representation vector $(\mathbf{h}_{\mathcal{G}})_k := \text{Q-Former}(\mathbf{q}_k)$. The query tokens share the same self-attention modules with the texts, but use different MLPs, allowing the Q-Former to be used for obtaining the representation of text sequence $\mathbf{h}_{d_{\mathcal{G}}} := \text{Q-Former}([\text{CLS}])$. Then we have $\ell_{\text{MolCA}} = -\ell_{\text{g2t}} - \ell_{\text{t2g}}$, where $\ell_{\text{g2t}} = \log \frac{\exp(\max_k \cos((\mathbf{h}_{\mathcal{G}})_k, \mathbf{h}_{d_{\mathcal{G}}})/\tau)}{\sum_{\tilde{d}_{\mathcal{G}} \neq d_{\mathcal{G}}} \exp(\max_k \cos((\mathbf{h}_{\mathcal{G}})_k, \mathbf{h}_{\tilde{d}_{\mathcal{G}}})/\tau)}$, and $\ell_{\text{t2g}} = \log \frac{\exp(\max_k \cos(\mathbf{h}_{d_{\mathcal{G}}}, (\mathbf{h}_{\mathcal{G}})_k)/\tau)}{\sum_{\tilde{\mathcal{G}} \neq \mathcal{G}} \exp(\max_k \cos(\mathbf{h}_{d_{\mathcal{G}}}, (\mathbf{h}_{\tilde{\mathcal{G}}})_k)/\tau)}$.

2) Discussion: Larger-Scale GNNs: GNNs integrate atomic and graph structural features for molecular representation learning [144]. Specifically, Text2Mol [121] utilizes the GCN [83] as its graph encoder and extracts unique identifiers for node features based on Morgan fingerprints [185]. MoMu [173], MoMu-v2 [172], MolFM [161], GIT-Mol [157], and MolCA [166] prefer GIN [188] as the backbone, as GIN has been proven to be as expressive and powerful as the Weisfeiler-Lehman graph isomorphism test. As described in Section II-B, there has been notable progress in making GNNs deeper, more generalizable, and more powerful since the proposal of the GCN [83] in 2016 and the GIN [188] in 2018. However, most reviewed works [157], [161], [166], [172], [173] are developed using the GIN [188] as a proof of concept for their approaches. These pretrained GINs feature five layers and 300 hidden dimensions. The scale of GNNs may be a bottleneck in learning semantic meaningful representation and there is a risk of over-reliance on one modality, neglecting the other. Therefore, for future large-scale GNN designs comparable to LLMs, scaling up the dimension size and adding deeper layers, may be considered. Besides, Transformer encoders [141] may also improve the expressive power of deep GNNs.

Generation Decoder with GNNs: GNNs are often not used as decoders for graph generation. The prevalent decoders are mostly text-based, generating linearized graph structures such as SMILES. These methods may be sensitive to the sequence order in the linearized graph. Generative diffusion models [201] on graphs could be utilized in future work to design generators with GNNs.

VII. RESOURCES AND APPLICATIONS

A. Datasets, Splitting and Evaluation

We summarize the datasets for three scenarios (namely pure graphs, text-attributed graphs, and text-paired graphs) and show them in Tables V, II, and III respectively.

1) Pure Graphs: In Table 5, we summarize the pure graph reasoning problems discussed in Section IV. Many problems are shared or revisited in different datasets due to their commonality. NLGraph [123], LLMtoGraph [124] and GUC [125] study a set of standard graph reasoning problems, including connectivity, shortest path, and graph diameter. GraphQA [130] benchmarks a similar set of problems but additionally describes the graphs in real-world scenarios to study the effect of graph grounding. LLM4DyG [127] focuses on reasoning tasks on temporally evolving graphs. Accuracy is the most common evaluation metric as they are primarily formulated as graph question-answering tasks.

2) Text-Attributed Graphs: We summarize the famous datasets for evaluating models on text-attributed graphs in Table II. The datasets are mostly from the academic, e-commerce, book, social media, and Wikipedia domains. The popular tasks to evaluate models on those datasets include node classification, link prediction, edge classification, regression, and recommendation. The evaluation metrics for node/edge classification include Accuracy, Macro-F1, and Micro-F1. For link prediction and recommendation evaluation, Mean

TABLE III
DATA COLLECTION IN SECTION VI FOR TEXT-CAPTIONED GRAPHS

Data	Date	Task	Size	Source & Notes
ChEMBL-2023 [184]	2023	Various	2.4M[2],20.3M[3]	Drug-like
PubChem [182]	2019	Various	96M[2],237M[3]	Biomedical
PC324K [166]	2023	PT, Cap.,	324K[1]	PubChem [182]
MolXPT-PT [168]	2023	PT	30M[2]	PubChem [182], PubMed, ChEBI [181]
ChE-bio [46]	2023	PT	365K[2]	ChEMBL [183]
ChE-phy [46]	2023	PT	365K[2]	ChEMBL [183]
ChE ZS [46]	2023	GC	91K[2]	ChEMBL [183]
PC223M [169]	2023	PT, Retr.	223M[1],2M[2],20K[3]	PubChem [182]
PCSTM [171]	2022	PT	281K[1]	PubChem [182]
PCdes [182]	2022	FT, Cap, Retr.	15K[1]	PubChem [182]
ChEBI-20 [121]	2021	FT., Retr., Gen., Cap.	33K[1]	PubChem [182], ChEBI [181]

"PT", "FT", "Cap.", "GC", "Retr.', and "Gen." refer to pretraining, finetuning, caption, graph classification, retrieval, and graph generation, respectively. The superscript for the size denotes # graph-text pairs[1], # graphs[2], # assays[3].

Reciprocal Rank (MRR), Normalized Discounted Cumulative Gain (NDCG), and Hit Ratio (Hit) usually serve as metrics. While evaluating model performance on regression tasks, people tend to adopt mean absolute errors (MAE) or root mean square error (RMSE).

3) Text-Paired Graphs: Table III shows text-paired graph datasets (including text-available and graph-only datasets). For *Data Splitting*, options include random splitting, source-based splitting, activity cliffs and scaffolds [195], and data balancing [142]. Graph classification usually adopts AUC [187] as the metrics, while regression uses MAE, RMSE, and R^2 [144]. For text generation evaluation, people tend to use the Bilingual Evaluation Understudy (BLEU) score; while for molecule generation evaluation, heuristic evaluation methods (based on factors including validity, novelty, and uniqueness) are adopted. However, it is worth noted that BLEU score is efficient but less accurate, while heuristic evaluation methods are problematic subject to unintended modes, such as the superfluous addition of carbon atoms in [196].

B. Open-Source Implementations

HuggingFace: HF Transformers[1] is the most popular Python library for Transformers-based language models. Besides, it also provides two additional packages: Datasets[2] for easily accessing and sharing datasets and Evaluate[3] for easily evaluating machine learning models and datasets.

Fairseq: Fairseq[4] is another open-source Python library for Transformers-based language models.

PyTorch Geometric: PyG[5] is an open-source Python library for graph machine learning. It packages more than 60 types of GNN, aggregation, and pooling layers.

Deep Graph Library: DGL[6] is another open-source Python library for graph machine learning.

RDKit: RDKit[7] is one of the most popular open-source cheminformatics software programs that facilitates various operations and visualizations for molecular graphs. It offers many useful APIs, such as the linearization implementation for molecular graphs, to convert them into easily stored SMILES and to convert these SMILES back into graphs.

C. Practical Applications

1) Scientific Discovery: Virtual Screening: It aims to search a library of unlabeled molecules to identify useful structures for a given task. Machine learning models could automatically screen out trivial candidates to accelerate this process. However, training accurate models is not easy since labeled molecules are limited in size and imbalanced in distribution [142]. There are many efforts to improve GNNs against data sparsity [142], [144], [191]. However, it is difficult for a model to generalize and understand in-depth domain knowledge that it has never been trained on. Texts could be complementary knowledge sources. Discovering task-related content from massive scientific papers and using them as instructions has great potential to design accurate GNNs in virtual screening [46].

Molecular Generation: Molecular generation and optimization is one fundamental goal for drug and material discovery. Scientific hypotheses of molecules [198], can be represented in the joint space of GNNs and LLMs. Then, one may search in the latent space for a better hypothesis that aligns with the text description (human requirements) and adheres to structural constraints like chemical validity. Chemical space has been found to contain more than 10^{60} molecules [197], which is beyond the capacity of exploration in wet lab experiments. Generating constrained candidates within relevant subspaces is a challenge [201] and promising, especially when incorporating textual conditions.

Synthesis Planning: Synthesis designs start from available molecules and involve planning a sequence of steps that can finally produce a desired chemical compound through a series of reactions [198]. This procedure includes a sequence of reactant molecules and reaction conditions. Both graphs and texts play

[1]https://huggingface.co/docs/transformers/index
[2]https://huggingface.co/docs/datasets/index
[3]https://huggingface.co/docs/evaluate/index
[4]https://github.com/facebookresearch/fairseq
[5]https://pytorch-geometric.readthedocs.io/en/latest/index.html
[6]https://www.dgl.ai/
[7]https://www.rdkit.org/docs/

important roles in this process. For example, graphs may represent the fundamental structure of molecules, while texts may describe the reaction conditions, additives, and solvents. LLMs can assist in the planning by suggesting possible synthesis paths directly or by serving as agents to operate on existing planning tools [145].

2) Computational Social Science: In computational social science, researchers are interested in modeling the behavior of people/users and discovering new knowledge that can be utilized to forecast the future. The behaviors of users and interactions between users can be modeled as graphs, where the nodes are associated with rich text information (e.g., user profile, messages, emails). We will show two example scenarios below.

E-commerce: In E-commerce platforms, there are many interactions (e.g., purchase, view) between users and products. For example, users can view or purchase products. In addition, the users, products, and their interactions are associated with rich text information. For instance, products have titles/descriptions and users can leave a review of products. In this case, we can construct a graph [101] where nodes are users and products, while edges are their interactions. Both nodes and edges are associated with text. It is important to utilize both the text information and the graph structure information (user behavior) to model users and items and solve complex downstream tasks (e.g., item recommendation [105], bundle recommendation [106], and product understanding [107]).

Social Media: In social media platforms, there are many users and they interact with each other through messages, emails, and so on. In this case, we can build a graph where nodes are users and edges are the interaction between users. There will be text associated with nodes (e.g., user profile) and edges (e.g., messages). Interesting research questions will be how to do joint text and graph structure modeling to deeply understand the users for friend recommendation [108], user analysis [109], community detection [110], and personalized response generation [96], [97].

3) Specific Domains: In many specific domains, text data are interconnected and lie in the format of graphs. The structure information on the graphs can be utilized to better understand the text unit and contribute to advanced problem-solving.

Academic Domain: In the academic domain, graphs [12] are constructed with papers as nodes and their relations (e.g., citation, authorship, etc) as edges. The representation learned for papers on such graphs can be utilized for paper recommendation [102], paper classification [103], and author identification [104].

Legal Domain: In the legal domain, opinions given by the judges always contain references to opinions given for previous cases. In such scenarios, people can construct a graph [98] based on the citation relations between opinions. The representations learned on such a graph with both text and structure information can be utilized for clause classification [99] and opinion recommendation [100].

Education Domain: In the education domain, we can construct a graph with coursework as nodes and their relations as edges. The model learned on such a graph can be utilized for knowledge tracing [135] and student performance prediction [136].

VIII. Future Directions

Better Benchmark Datasets: Most pure graph benchmarks evaluate LLMs' reasoning ability on homogeneous graphs but do not include evaluations on heterogeneous or spatial-temporal graphs. For text-attributed graphs, as summarized in Table II, most benchmark datasets are from academic domains and e-commerce domains. However, in the real world, text-attributed graphs are ubiquitous across multiple domains (e.g., legal and health). More diverse datasets are needed to comprehensively evaluate LLMs on real-world scenarios. For text-paired graphs, as summarized in Table III, there is a lack of comprehensive datasets covering various machine learning tasks in chemistry. Although a massive number of scientific papers are available, preprocessing them into a ready-to-use format and pairing them with specific molecular graph data points of interest remains a cumbersome and challenging task. Besides, we could investigate graph-text pairs in 3D space, where each molecule may be associated with atomic coordinates [137].

Broader Task Space with LLMs: More comprehensive studies on the performance of LLMs for graph tasks hold promise for the future. While LLMs as encoder approaches have been explored for text-attributed graphs, their application to text-captioned molecular graphs remains underexplored. Promising directions include using LLMs for data augmentation and knowledge distillation to design domain-specific GNNs for various text-paired graph tasks. Furthermore, although graph generation has been approached in text-paired graphs, it remains an open problem for text-attributed graphs (i.e., how to conduct joint text and graph structure generation)

Efficienct LLMs on Graphs: While LLMs have shown a strong capability to learn on graphs, they suffer from inefficiency in graph linearization and model optimization. On one hand, as discussed in Sections V-A1 and VI-A1, many methods rely on transferring graphs into sequences that can be inputted into LLMs. However, the length of the transferred sequence will increase significantly as the size of the graph increases. This poses challenges since LLMs always have a maximum sequence input length and a long input sequence will lead to higher time and memory complexity. On the other hand, optimizing LLMs itself is computationally expensive. Although some general efficient tuning methods such as LoRA are proposed, there is a lack of discussion on graph-aware LLM efficient tuning methods.

Generalizable and Robust LLMs on Graphs: Another interesting direction is to explore the generalizability and robustness of LLMs on graphs. Generalizability refers to having the ability to transfer the knowledge learned from one domain graph to another; while robustness denotes having consistent prediction regarding obfuscations and attacks. Although LLMs have demonstrated their strong generalizability in processing text, they still suffer from robustness and hallucination issues, which are to be solved for graph data modeling as well.

Multi-Modal Foundation Models: One open question is, "Should we use one foundation model to unify different modalities, and how?" The modalities can include texts, graphs, and even images. For instance, molecules can be represented as

graphs, described as texts, and photographed as images; products can be treated as nodes in a graph, associated with a title/description, and combined with an image. Designing a model that can conduct joint encoding for all modalities will be useful but challenging. Furthermore, there has always been tension between building a unified foundational model and customizing model architectures for different domains. It is thus intriguing to ask whether a unified architecture will suit different data types, or if tailoring model designs according to domains will be necessary. Correctly answering this question can save economic and intellectual resources from unnecessary attempts and also shed light on a deeper understanding of graph-related tasks.

LLMs as Dynamic Agents on Graphs: Although LLMs have shown their advanced capability in generating text, one-pass generation of LLMs suffers from hallucination and misinformation issues due to the lack of accurate parametric knowledge. Simply augmenting retrieved knowledge in context is also bottlenecked by the capacity of the retriever. In many real-world scenarios, graphs such as academic networks, and Wikipedia are dynamically looked up by humans for knowledge-guided reasoning. Simulating such a role of dynamic agents can help LLMs more accurately retrieve relevant information via multi-hop reasoning, thereby correcting their answers and alleviating hallucinations.

IX. Conclusion

In this paper, we provide a comprehensive review of large language models on graphs. We first categorize graph scenarios where LMs can be adopted and summarize the large language models on graph techniques. We then provide a thorough review, analysis, and comparison of methods within each scenario. Furthermore, we summarize available datasets, open-source codebases, and multiple applications. Finally, we suggest future directions for large language models on graphs.

Acknowledgment

Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily represent the views, either expressed or implied, of DARPA or the U.S. Government.

References

[1] W. Yang et al., "End-to-end open-domain question answering with bertserini," in *Proc. Conf. North Amer. Assoc. Comput. Linguistics*, 2019, pp. 72–77.
[2] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2019, pp. 3728–3738.
[3] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S.R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *Proc. Int. Conf. Learn. Representations*, 2018.
[4] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2019, pp. 3980–3990.
[5] J. Wei et al., "Emergent abilities of large language models," *Trans. Mach. Learn. Res.*, 2022.
[6] H. Nagamochi and T. Ibaraki, *Algorithmic Aspects of Graph Connectivity*, Cambridge, U.K.: Cambridge Univ. Press, 2018.
[7] A. V. Goldberg and C. Harrelson, "Computing the shortest path: A search meets graph theory," in *Proc. 16th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2005, pp. 156–165.
[8] Z. Sun, H. Wang, H. Wang, B. Shao, and J. Li, "Efficient subgraph matching on billion node graphs," 2012, *arXiv:1205.6691*.
[9] Z. Chen et al., "Exploring the potential of large language models (LLMS) in learning on graphs," 2023, *arXiv:2307.03393*.
[10] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Inf. Retrieval*, vol. 3, pp. 127–163, 2000.
[11] C. L. Giles, K. D. Bollacker, and S. Lawrence, "CiteSeer: An automatic citation indexing system," in *Proc. 3rd ACM Conf. Digit. Libraries*, 1998, pp. 89–98.
[12] K. Wang, Z. Shen, C. Huang, C. H. Wu, Y. Dong, and A. Kanakia, "Microsoft academic graph: When experts are not enough," *Quantitative Sci. Stud.*, vol. 1, no. 1, pp. 396–413, 2020.
[13] Y. Zhang, B. Jin, Q. Zhu, Y. Meng, and J. Han, "The effect of metadata on scientific literature tagging: A cross-field cross-model study," in *Proc. Int. Conf. World Wide Web*, 2023, pp. 1626–1637.
[14] M. Wan and J. McAuley, "Item recommendation on monotonic behavior chains," *RecSys*, 2018, pp. 86–94.
[15] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *Proc. Conf. Empir. Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process.*, 2019, pp. 188–197.
[16] P. Sen, G. Namata, M. Bilgic, L. Getoor, and B. Galligher, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, pp. 93–93, 2008.
[17] X. Wang et al., "KEPLER: A unified model for knowledge embedding and pre-trained language representation," *Trans. Assoc. Comput. Linguistics*, vol. 9, pp. 176–194, 2021.
[18] L. Liu, B. Du, H. Ji, C. Zhai, and H. Tong, "Neural-answering logical queries on knowledge graphs," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*., 2021, pp. 1087–1097.
[19] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. P. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.
[20] J. Liu et al., "Towards graph foundation models: A survey and beyond," 2023, *arXiv:2310.11829*.
[21] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," 2023, *arXiv:2306.08302*.
[22] Y. Wang, H. Le, A. D. Gotmare, N. D. Bui, J. Li, and S. C. Hoi, "Codet5 : Open code large language models for code understanding and generation," 2023, *arXiv:2305.07922*.
[23] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-Training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Assoc. Comput. Linguistics*, 2019, pp. 4171–4186.
[24] Y. Liu et al., "RoBERTa: A robustly optimized bert pretraining approach," 2019, *arXiv: 1907.11692*.
[25] I. Beltagy, K. Lo, and A. Cohan, "SciBERT: A pretrained language model for scientific text," 2019, *arXiv: 1903.10676*.
[26] T. Brown et al., "Language models are few-shot learners," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020.
[27] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 5754–5764.
[28] M. Lewis et al., "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proc. Conf. Assoc. Comput. Linguistics*, 2020, pp. 7871–7880.
[29] C. Raffel et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, pp. 140:1–140:67, 2020.
[30] M. Yasunaga, J. Leskovec, and P. Liang, "LinkBERT: Pretraining language models with document links," in *Proc. Conf. Assoc. Comput. Linguistics*, 2022, pp. 8003–8016.
[31] B. Jin et al., "Patton: Language model pretraining on text-rich networks," in *Proc. Conf. Assoc. Comput. Linguistics*, 2023, pp. 7005–7020.
[32] T. Zou, L. Yu, Y. Huang, L. Sun, and B. Du, "Pretraining language models with text-attributed heterogeneous graphs," 2023, *arXiv:2310.12580*.
[33] K. Song, X. Tan, T. Qin, and J. Lu, "MPNet: Masked and permuted pre-training for language understanding," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 16857–16867.
[34] K. Duan et al., "SimTeG: A frustratingly simple approach improves textual graph learning," 2023, *arXiv:2308.02565*.

[35] E. Kasneci et al., "ChatGPT for good? On opportunities and challenges of large language models for education," *Learn. Individual Differences*, vol. 103, 2023, Art. no. 102274.
[36] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2021, pp. 3045–3059.
[37] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *Proc. Conf. Assoc. Comput. Linguistics*, 2021, pp. 4582–4597.
[38] N. Houlsby et al., "Parameter-efficient transfer learning for NLP," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2790–2799.
[39] E. J. Hu et al., "Lora: Low-rank adaptation of large language models," in *Proc. Int. Conf. Learn. Representations*, 2022.
[40] Y. Tian et al., "Graph neural prompting with large language models," 2023, *arXiv:2309.15427*.
[41] Z. Chai et al., "GraphLLM: Boosting graph reasoning ability of large language model," 2023, *arXiv:2310.05845*.
[42] J. Wei et al., "Finetuned language models are zero-shot learners," in *Proc. Int. Conf. Learn. Representations*, 2022.
[43] V. Sanh et al., "Multitask prompted training enables zero-shot task generalization," in *Proc. Int. Conf. Learn. Representations*, 2022.
[44] J. Tang et al., "GraphGPT: Graph instruction tuning for large language models," 2023, *arXiv:2310.13023*.
[45] R. Ye, C. Zhang, R. Wang, S. Xu, and Y. Zhang, "Natural language is all a graph needs," 2023, *arXiv:2308.07134*.
[46] H. Zhao et al., "GIMLET: A unified graph-text model for instruction-based molecule zero-shot learning," 2023, *arXiv:2306.13089*.
[47] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 24824–24837.
[48] S. Yao et al., "Tree of thoughts: Deliberate problem solving with large language models," 2023, *arXiv:2305.10601*.
[49] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, L. Gianinazzi, and J. Gajda, "Graph of thoughts: Solving elaborate problems with large language models," 2023, *arXiv:2308.09687*.
[50] A. Cohan, S. Feldman, I. Beltagy, D. Downey, and D.S. Weld, "Specter: Document-level representation learning using citation-informed transformers," in *Proc. Conf. Assoc. Comput. Linguistics*, 2020, pp. 2270–2282.
[51] M. Ostendorff, N. Rethmeier, I. Augenstein, B. Gipp, and G. Rehm, "Neighborhood contrastive learning for scientific document representations with citation embeddings," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2022, pp. 11670–11688.
[52] W. Brannon et al., "ConGraT: Self-supervised contrastive pretraining for joint graph and text embeddings," 2023, *arXiv:2305.14321*.
[53] J. Zhu, X. Song, V. N. Ioannidis, D. Koutra, and C. Faloutsos, "TouchUp-G: Improving feature representation through graph-centric finetuning," 2023, *arXiv:2309.13885*.
[54] Y. Li, K. Ding, and K. Lee, "GRENADE: Graph-centric language model for self-supervised representation learning on text-attributed graphs," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2023, pp. 2745–2757.
[55] X. Zhang, Y. Malkov, O. Florez, S. Park, B. McWilliams, and J. Han, "TwHIN-BERT: A socially-enriched pre-trained language model for multilingual tweet representations at twitter," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2023, pp. 5597–5607.
[56] X. Zhang, C. Zhang, X. L. Dong, J. Shang, and J. Han, "Minimally-supervised structure-rich text categorization via learning on text-rich networks," in *Proc. Int. Conf. World Wide Web*, 2021, pp. 3258–3268.
[57] E. Chien et al., "Node feature extraction by self-supervised multi-scale neighborhood prediction," in *Proc. Int. Conf. Learn. Representations*, 2022.
[58] Y. Zhang et al., "Metadata-induced contrastive learning for zero-shot multi-label text classification," in *Proc. Int. Conf. World Wide Web*, 2022, pp. 3162–3173.
[59] T. A. Dinh, J. D. Boef, J. Cornelisse, and P. Groth, "E2EG: End-to-end node classification using graph topology and text-based node attributes," 2022, *arXiv:2208.04609*.
[60] Y. Tan, Z. Zhou, H. Lv, W. Liu, and C. Yang, "WalkLM: A uniform language model fine-tuning framework for attributed graph embedding," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2023, pp. 13308–13325.
[61] J. Zhao et al., "Learning on large-scale text-attributed graphs via variational inference," in *Proc. Int. Conf. Learn. Representations*, 2023.
[62] Z. Wen and Y. Fang, "Augmenting low-resource text classification with graph-grounded pre-training and prompting," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2023, pp. 506–516.
[63] Z. Chen et al., "Label-free node classification on graphs with large language models (LLMS)," 2023, *arXiv:2310.04668*.
[64] J. Zhao et al., "GraphText: Graph reasoning in text space," 2023, *arXiv:2310.01089*.
[65] Y. Meng, S. Zong, X. Li, X. Sun, and T. Zhang,, "GNN-LM: Language modeling based on global contexts via GNN," in *Proc. Int. Conf. Learn. Representations*, 2022.
[66] X. Zhang et al., "GreaseLM: Graph reasoning enhanced language models for question answering," in *Proc. Int. Conf. Learn. Representations*, 2022.
[67] V. N. Ioannidis et al., "Efficient and effective training of language and graph neural network models," in *Proc. AAAI Conf. Artif. Intell.*, 2023.
[68] C. Mavromatis et al., "Train your own GNN teacher: Graph-aware distillation on textual graphs," in *Proc. Mach. Learn. Knowl. Discov. Databases: Res. Track*, 2023, pp. 157–173.
[69] X. He, X. Bresson, T. Laurent, and B. Hooi, "Explanations as features: LLM-based features for text-attributed graphs," 2023, *arXiv:2305.19523*.
[70] J. Yu, Y. Ren, C. Gong, J. Tan, X. Li, and X. Zhang, "Empower text-attributed graphs learning with large language models (LLMs)," 2023, *arXiv:2310.09872*.
[71] J. Yang et al., "GraphFormers: GNN-nested transformers for representation learning on textual graph," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 28798–28810.
[72] B. Jin, Y. Zhang, Q. Zhu, and J. Han, "Heterformer: Transformer-based deep node representation learning on heterogeneous text-rich networks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2023, pp. 1020–1031.
[73] B. Jin, Y. Zhang, Y. Meng, and J. Han, "Edgeformers: Graph-empowered transformers for representation learning on textual-edge networks," in *Proc. Int. Conf. Learn. Representations*, 2023.
[74] B. Jin, W. Zhang, Y. Zhang, Y. Meng, H. Zhao, and J. Han, "Learning multiplex embeddings on text-rich networks with one text encoder," 2023, *arXiv:2310.06684*.
[75] Y. Qin, X. Wang, Z. Zhang, and W. Zhu, "Disentangled representation learning with large language models for text-attributed graphs," 2023, *arXiv:2310.18152*.
[76] J. Zhu et al., "TextGNN: Improving text encoder via graph neural network in sponsored search," in *Proc. Int. Conf. World Wide Web*, 2021, pp. 2848–2857.
[77] C. Li et al., "Adsgnn: Behavior-graph augmented relevance modeling in sponsored search," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2021, pp. 223–232.
[78] J. Zhang, W.C. Chang, H. F. Yu, and I. Dhillon, "Fast multi-resolution transformer fine-tuning for extreme multi-label text classification," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 7267–7280.
[79] H. Xie et al., "Graph-aware language model pre-training on a large graph corpus can help multiple graph applications," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2023, pp. 5270–5281.
[80] M. Yasunaga et al., "Deep bidirectional language-knowledge graph pre-training," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 37309–37323.
[81] J. Huang, X. Zhang, Q. Mei, and J. Ma, "Can LLMs effectively leverage graph structural information: When and why," 2023, *arXiv:2309.16595*.
[82] X. Jin, B. Vinzamuri, S. Venkatapathy, H. Ji, and P. Natarajan, "Adversarial robustness for large language NER models using disentanglement and word attributions," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2023, pp. 12437–12450.
[83] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017.
[84] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.
[85] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018.
[86] S. Zhang, Y. Liu, and Y. Sun, "Graph-less neural networks: Teaching old MLPs new tricks via distillation," in *Proc. Int. Conf. Learn. Representations*, 2022.
[87] M. Liu, H. Gao, and S. Ji, "Towards deeper graph neural networks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 338–348.
[88] Y. Meng, J. Huang, Y. Zhang, and J. Han, "Generating training data with language models: Towards zero-shot language understanding," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 462–477.

[89] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "PathSim: Meta path-based top-k similarity search in heterogeneous information networks," in *Proc. VLDB Endowment*, vol. 4, pp. 992–1003, 2011.

[90] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2023.

[91] C. Park, D. Kim, J. Han, and H. Yu, "Unsupervised attributed multiplex network embedding," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 5371–5378.

[92] A. Vaswani et al., "Attention is all you need," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017.

[93] T.H. Haveliwala, "Topic-sensitive pagerank," in *Proc. Int. Conf. World Wide Web*, 2002, pp. 517–526.

[94] A. V. D. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018, *arXiv: 1807.03748*.

[95] A. Radford et al., "Learning transferable visual models from natural language supervision," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8748–8763.

[96] C. Sun et al., "Decoding the silent majority: Inducing belief augmented social graph with large language model for response forecasting," 2023, *arXiv:2310.13297*.

[97] C. Sun, J. Li, H.P. Chan, C. Zhai, and H. Ji, "Measuring the effect of influential messages on varying personas," in *Proc. Conf. Assoc. Comput. Linguistics*, 2023, pp. 554–562.

[98] R. Whalen, "Legal networks: The promises and challenges of legal network analysis," *Michigan State Law Rev.*, 2016, Art. no. 539.

[99] A. Friedrich, and A. Palmer, and M. Pinkal, "Situation entity types: Automatic classification of clause-level aspect," in *Proc. Conf. Assoc. Comput. Linguistics*, 2016, pp. 1757–1768.

[100] N. Guha et al., "LegalBench: A collaboratively built benchmark for measuring legal reasoning in large language models," 2023, *arXiv:2308.11462*.

[101] Y. Lin et al., "Personalized entity resolution with dynamic heterogeneous knowledge graph representations," 2021, *arXiv:2104.02667*.

[102] X. Bai, M. Wang, I. Lee, Z. Yang, X. Kong, and F. Xia, "Scientific paper recommendation: A survey," *IEEE Access*, vol. 7, pp. 9324–9339, 2019.

[103] S. Chowdhury and M. P. Schoen, "Research paper classification using supervised machine learning techniques," in *Proc. Int. Eng. Technol. Comput.*, 2020, pp. 1–6.

[104] D. Madigan, A. Genkin, D.D. Lewis, S. Argamon, D. Fradkin, and L. Ye, "Author identification on the large scale," in *Proc. Meeting Classification Soc. North Amer.*, 2005.

[105] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 639–648.

[106] J. Chang, C. Gao, X. He, D. Jin, and Y. Li, "Bundle recommendation with graph convolutional networks," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 1673–1676.

[107] H. Xu, B. Liu, L. Shu, and P. Yu, "Open-world learning and application to product classification," in *Proc. Proc. Int. Conf. World Wide Web*, 2019, pp. 3413–3419.

[108] L. Chen, Y. Xie, Z. Zheng, H. Zheng, and J. Xie, "Friend recommendation based on multi-social graph convolutional network," *IEEE Access*, vol. 8, pp. 43618–43629, 2020.

[109] G. Wang, X. Zhang, S. Tang, and H. Zheng, "Unsupervised clickstream clustering for user behavior analysis," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, 2016, pp. 225–236.

[110] O. Shchur and S. Günnemann, "Overlapping community detection with graph neural networks," 2019, *arXiv: 1909.12201*.

[111] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shotreasoners," *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 22199–22213, 2022.

[112] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 22199–22213.

[113] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 24824–24837.

[114] A. Radford, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, 2019, Art. no. 9.

[115] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite bert for self-supervised learning of language representations," in *Proc. Int. Conf. Learn. Representations*, 2019.

[116] K. Clark, M. T. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: Pre-training text encoders as discriminators rather than generators," in *Proc. Int. Conf. Learn. Representations*, 2019.

[117] S. Bubeck et al., "Sparks of artificial general intelligence: Early experiments with GPT-4," 2023, *arXiv:2303.12712*.

[118] H. Touvron et al., "Llama 2: Open foundation and fine-tuned chat models," 2023, *arXiv:2307.09288*.

[119] A. Q. Jiang et al., "Mistral 7B," 2023, *arXiv:2310.06825*.

[120] J.B. Alayrac et al., "Flamingo: A visual language model for few-shot learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 23716–23736.

[121] C. Edwards, C. Zhai, and H. Ji, "Text2Mol: Cross-modal molecule retrieval with natural language queries," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2022, pp. 595–607.

[122] C. Edwards, T. Lai, K. Ros, G. Honke, and H. Ji, "Translation between molecules and natural language," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2022, pp. 375–413.

[123] H. Wang, S. Feng, T. He, Z. Tan, X. Han, and Y. Tsvetkov, "Can language models solve graph problems in natural language?," 2023, *arXiv:2305.10037*.

[124] C. Liu and B. Wu, "Evaluating large language models on graphs: Performance insights and comparative analysis," 2023, *arXiv:2308.11224*, 2023.

[125] J. Guo, L. Du, and H. Liu, "GPT4Graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking," 2023, *arXiv:2305.15066*.

[126] J. Zhang, "Graph-ToolFormer: To empower LLMs with graph reasoning ability via prompt augmented by ChatGPT," 2023, *arXiv:2304.11116*.

[127] Z. Zhang et al., "LLM4DyG: Can large language models solve problems on dynamic graphs?," 2023, *arXiv:2310.17110*.

[128] L. Luo, Y.F. Li, G. Haffari, and S. Pan, "Reasoning on graphs: Faithful and interpretable large language model reasoning," 2023, *arXiv:2310.01061*.

[129] J. Jiang, K. Zhou, Z. Dong, K. Ye, W. X. Zhao, and J. R. Wen, "StructGPT: A general framework for large language model to reason over structured data," 2023, *arXiv:2305.09645*.

[130] B. Fatemi, J. Halcrow, and B. Perozzi, "Talk like a graph: Encoding graphs for large language models," 2023, *arXiv:2310.04560*.

[131] J. Sun et al., "Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph," 2023, *arXiv:2307.07697*.

[132] D. Z. Chen, "Developing algorithms and software for geometric path planning problems," *ACM Comput. Surv*, vol. 28, pp. 18–es, 1996, doi: 10.1145/242224.242246.

[133] A. Iqbal, M. Hossain, and A. Ebna, "Airline scheduling with max flow algorithm," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018.

[134] L. Jiang, X. Zang, I. I. Alghoul, X. Fang, J. Dong, and C. Liang, "Scheduling the covering delivery problem in last mile delivery," *Expert Syst. Appl.*, vol. 187, 2022, Art. no. 115894.

[135] H. Nakagawa, Y. Iwasawa, and Y. Matsuo, "Graph-based knowledge tracing: Modeling student proficiency using graph neural network," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell.*, 2019, pp. 156–163.

[136] H. Li, H. Wei, Y. Wang, Y. Song, and H. Qu, "Peer-inspired student performance prediction in interactive online question pools with graph neural network," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, 2020, pp. 2589–2596.

[137] X. Zhang et al., "Artificial intelligence for science in quantum, atomistic, and continuum systems," 2023, *arXiv:2307.08423*.

[138] T. K. Rusch, M. M. Bronstein, and S. Mishra, "A survey on oversmoothing in graph neural networks," 2023, *arXiv:2303.10993*.

[139] J. Topping, F. D. Giovanni, B. P. Chamberlain, X. Dong, and M. M. Bronstein, Understanding over-squashing and bottlenecks on graphs via curvature," 2021, *arXiv:2111.14522*.

[140] C. Ying et al., "Do transformers really perform badly for graph representation?," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 28877–28888.

[141] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, "Recipe for a general, powerful, scalable graph transformer," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 14501–14515.

[142] G. Liu, T. Zhao, E. Inae, T. Luo, and M. Jiang, "Semi-supervised graph imbalanced regression," 2023, *arXiv:2305.12087*.

[143] Q. Wu, W. Zhao, Z. Li, D. P. Wipf, and J. Yan, "Nodeformer: A scalable graph structure learning transformer for node classification," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 27387–401.

[144] G. Liu, T. Zhao, J. Xu, T. Luo, and M. Jiang, "Graph rationalization with environment-based augmentations," *ACM Trans. Knowl. Discov. Data*, vol. 18, no. 4, pp. 1–23, 2022.

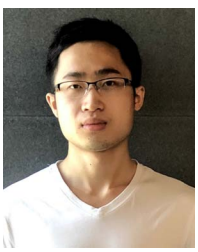
[145] A. M. Bran, S. Cox, A. D. White, and P. Schwaller, "ChemCrow: Augmenting large-language models with chemistry tools," 2023, *arXiv:2304.05376*.

[146] K. Riesen and H. Bunke, "IAM graph database repository for graph based pattern recognition and machine learning," in *Proc. Struct., Syntactic, Statist. Pattern Recognit. Joint IAPR Workshop*, 2008, pp. 287–297.
[147] D. Weininger, "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules," *J. Chem. Inf. Comput. Sci.*, vol. 28, no. 1, pp. 31–36, 1988.
[148] S. Heller, A. McNaught, S. Stein, D. Tchekhovskoi, and I. Pletnev, "InChI-the worldwide chemical structure identifier standard," *J. Cheminformatics*, vol. 5, no. 1, pp. 1–9, 2013.
[149] N. O'Boyle and A. Dalke, "DeepSMILES: An adaptation of SMILES for use in machine-learning of chemical structures," 2018.
[150] M. Krenn, F. Häse, A. Nigam, P. Friederich, and A. Aspuru-Guzik, "Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation," *Mach. Learn.: Sci. Technol.*, vol. 1, no. 4, 2020, Art. no. 045024.
[151] E. J. Bjerrum, "SMILES enumeration as data augmentation for neural network modeling of molecules," 2017, *arXiv: 1703.07076*.
[152] J. Arús-Pous et al., "Randomized SMILES strings improve the quality of molecular generative models," *J. Cheminformatics*, vol. 11, no. 1, pp. 1–13, 2019.
[153] I. V. Tetko, P. Karpov, E. Bruno, T. B. Kimber, and G. Godin, "Augmentation is what you need!," in *Proc. Int. Conf. Artif. Neural Netw.*, Springer International Publishing, 2019, pp. 831–835.
[154] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing, in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2018, pp. 66–71.
[155] R. Irwin, S. Dimitriadis, J. He, and E. J. Bjerrum, "Chemformer: A pre-trained transformer for computational chemistry," *Mach. Learn. Sci. Technol.*, vol. 3, no. 1, 2022, Art. no. 015022.
[156] Y. Shi, A. Zhang, E. Zhang, Z. Liu, and X. Wang, "ReLM: Leveraging language models for enhanced chemical reaction prediction," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2023, pp. 5506–5520.
[157] P. Liu, Y. Ren, and Z. Ren, "GIT-Mol: A multi-modal large language model for molecular science with graph, image, and text," 2023, *arXiv:2308.06911*.
[158] J. Ock, C. Guntuboina, and A. B. Farimani, "Catalyst property prediction with CatBERTa: Unveiling feature exploration strategies through large language models," 2023, *arXiv:2309.00563*.
[159] Y. Fang et al., "Mol-instructions: A large-scale biomolecular instruction dataset for large language models," 2023, *arXiv:2306.08018*.
[160] H. Abdine, M. Chatzianastasis, C. Bouyioukos, and M. Vazirgiannis, "Prot2Text: Multimodal protein's function generation with GNNs and transformers," 2023, *arXiv:2307.14367*.
[161] Y. Luo, K. Yang, M. Hong, X. Liu, and Z. Nie, "MolFM: A multimodal molecular foundation model," 2023, *arXiv:2307.09484*.
[162] C. Qian, H. Tang, Z. Yang, H. Liang, and Y. Liu, "Can large language models empower molecular property prediction?," 2023, *arXiv:2307.07443*.
[163] J. Born and M. Manica, "Regression transformer enables concurrent sequence regression and generation for molecular language modelling," *Nature Mach. Intell.*, vol. 5, no. 4, pp. 432–444, 2023.
[164] J. Li et al., "Empowering molecule discovery for molecule-caption translation with large language models: A ChatGPT perspective," 2023, *arXiv:2306.06615*.
[165] Z. Zeng, B. Yin, S. Wang, J. Liu, C. Yang, and H. Yao, "Interactive molecular discovery with natural language," 2023, *arXiv:2306.11976*.
[166] Z. Liu et al., "MolCA: Molecular graph-language modeling with cross-modal projector and uni-modal adapter," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2023, pp. 15623–15638.
[167] T. Guo et al., "What indeed can GPT models do in chemistry? A comprehensive benchmark on eight tasks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2023.
[168] Z. Liu, W. Zhang, Y. Xia, L. Wu, S. Xie, and T. Qin, "MolXPT: Wrapping molecules with text for generative pre-training," in *Proc. Conf. Assoc. Comput. Linguistics*, 2023, pp. 1606–1616.
[169] P. Seidl, A. Vall, S. Hochreiter, and G. Klambauer, "Enhancing activity prediction models in drug discovery with the ability to understand human language," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 30458–30490.
[170] D. Christofidellis, G. Giannone, J. Born, O. Winther, T. Laino, and M. Manica, "Unifying molecular and textual representations via multi-task language modelling," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 6140–6157.
[171] S. Liu et al., "A multi-modal molecule structure-text model for text-based retrieval and editing," *Nature Mach. Intell.*, vol. 5, pp. 1447–1457, 2023.
[172] R. Lacombe, A. Gaut, J. He, D. Lüdeke, and K. Pistunova, "Extracting molecular properties from natural language with multimodal contrastive learning," in *Proc. Int. Conf. Mach. Learn. Workshop Comput. Biol.*, 2023.
[173] B. Su et al., "A molecular multimodal foundation model associating molecule graphs with natural language," 2022, *arXiv:2209.05481*.
[174] Z. Zeng, Y. Yao, Z. Liu, and M. Sun, "A deep-learning system bridging molecule structure and biomedical text with comprehension comparable to human professionals," *Nature Commun.*, vol. 13, 2022, Art. no. 862.
[175] M. Iwayama, S. Wu, C. Liu, and R. Yoshida, "Functional output regression for machine learning in materials science," *J. Chem. Inf. Model.*, vol. 62, no. 20, pp. 4837–4851, 2022.
[176] V. Bagal, R. Aggarwal, P. K. Vinod, and U. D. Priyakumar, "MolGPT: Molecular generation using a transformer-decoder model," *J. Chem. Inf. Model.*, vol. 62, no. 9, pp. 2064–2076, 2021.
[177] R. Taylor, M. Kardas, G. Cucurull, T. Scialom, and A. Hartshorn, "Galactica: A large language model for science," 2022, *arXiv:2211.09085*.
[178] S. Wang, Y. Guo, Y. Wang, H. Sun, and J. Huang, "Smiles-BERT: Large scale unsupervised pre-training for molecular property prediction," in *Proc. ACM Int. Conf. Bioinf. Comput. Biol. Health Inform.*, 2019, pp. 429–436.
[179] J. Lee et al., "BioBERT: A pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.
[180] R. Ma and T. Luo, "PI1M: A benchmark database for polymer informatics," *J. Chem. Inf. Model.*, vol. 60, pp. 4684–4690, 2020.
[181] J. Hastings et al., "ChEBI in 2016: Improved services and an expanding collection of metabolites," *Nucleic acids Res.*, vol. 44, pp. D1214–D1219, 2016.
[182] S. Kim et al., "PubChem 2019 update: Improved access to chemical data," *Nucleic Acids Res.*, vol. 47, no. D1, pp. D1102–D1109, 2019.
[183] A. Gaulton et al., "ChEMBL: A large-scale bioactivity database for drug discovery," *Nucleic Acids Res.*, vol. 40, pp. D1100–D1107, 2012.
[184] B. Zdrazil et al., "The ChEMBL database in 2023: A drug discovery platform spanning multiple bioactivity data types and time periods," *Nucleic Acids Res.*, vol. 53, pp. D1180–D1192, 2024.
[185] C. L. Mellor et al., "Molecular fingerprint-derived similarity measures for toxicological read-across: Recommendations for optimal use," *Regulatory Toxicol. Pharmacol.*, 2019, pp. 121–134.
[186] M. Krenn et al., "SELFIES and the future of molecular string representations," *Patterns*, vol. 3, no. 10, 2022, Art. no. 100588.
[187] W. Hu et al., "Open graph benchmark: Datasets for machine learning on graphs," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 22118–22133.
[188] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *Proc. Int. Conf. Learn. Representations*, 2019.
[189] J. Li, D. Li, S. Savarese, and S. Hoi, "BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," *arXiv:2301.12597*.
[190] C. Zang and F. Wang, "Moflow: An invertible flow model for generating molecular graphs," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 617–626.
[191] G. Liu, E. Inae, T. Zhao, J. Xu, T. Luo, and M. Jiang, "Data-centric learning from unlabeled graphs with diffusion model," 2023, *arXiv:2303.10108*.
[192] Y. Wang, N. Lipka, R. A. Rossi, A. Siu, R. Zhang, and T. Derr, "Knowledge graph prompting for multi-document question answering," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2024, pp. 19206–19214.
[193] Z. Guo, W. Yu, C. Zhang, M. Jiang, and N.V. Chawla, "GraSeq: Graph and sequence fusion learning for molecular property prediction," in *Proc. Int. Conf. Inf. Knowl. Manage.*, 2020, pp. 435–443.
[194] W. Yu, C. Zhu, L. Qin, Z. Zhang, T. Zhao, and M. Jiang, "Diversifying content generation for commonsense reasoning with mixture of knowledge graph experts," in *Proc. Conf. Assoc. Comput. Linguistics Findings*, 2022, pp. 1–11.
[195] J. Deng, Z. Yang, H. Wang, I. Ojima, D. Samaras, and F. Wang, "A systematic study of key elements underlying molecular property prediction," *Nature Commun.*, vol. 14, no. 1, 2023, Art. no. 6395.
[196] P. Renz, D. Van Rompaey, J. K. Wegner, S. Hochreiter, and G. Klambauer, "On failure modes in molecule generation and optimization," *Drug Discov. Today, Technol.*, vol. 32, pp. 55–63, 2019.
[197] J. L. Reymond, "The chemical space project," *Accounts Chem. Res.*, vol. 48, no. 3, pp. 722–730, 2015.
[198] H. Wang et al., "Scientific discovery in the age of artificial intelligence," *Nature*, vol. 620, no. 7972, pp. 47–60, 2023.

[199] H. Wang et al., "Chemical-reaction-aware molecule representation learning," 2021, *arXiv:2109.09888*.
[200] T. M. Lai, C. Zhai, and H. Ji, "KEBLM: Knowledge-enhanced biomedical language models," *J. Biomed. Informat.*, vol. 143, 2023, Art. no. 104392.
[201] G. Liu, J. Xu, T. Luo, and M. Jiang, "Inverse molecular design with multi-conditional diffusion guidance," 2024, *arXiv:2401.13858*.
[202] M. Li et al., "The future is not one-dimensional: Complex event schema induction by graph modeling for event prediction," 2021, *arXiv:2104.06344*.
[203] Q. Mao, Z. Liu, C. Liu, Z. Li, and J. Sun, "Advancing graph representation learning with large language models: A comprehensive survey of techniques," 2024, *arXiv:2402.05952*.
[204] Y. Li et al., "A survey of graph meets large language model: Progress and future directions," 2023, *arXiv:2311.12399*.
[205] S. Griffith, K. Subramanian, J. Scholz, C. L. Isbell, and A. L. Thomaz, "Policy shaping: Integrating human feedback with reinforcement learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 2625–2633.
[206] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv: 1707.06347*.
[207] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2024, pp. 53728–53741.
[208] W.X. Zhao et al., "A survey of large language models," 2023, *arXiv:2303.18223*.
[209] C. Han, Q. Wang, W. Xiong, Y. Chen, H. Ji, and S. Wang, "LM-infinite: Simple on-the-fly length generalization for large language models," 2023, *arXiv:2308.16137*.
[210] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, 2024, Art. no. 127063.

Bowen Jin received the BS degree from Tsinghua University in 2021. He is currently working toward the PhD degree in computer science with the University of Illinois at Urbana-Champaign, advised by Prof. Jiawei Han. His research focuses on large language models, information networks, and data/text mining, with their applications in information retrieval and knowledge discovery. He has published first-authored papers in SIGIR, ICLR, ACL, and KDD. He receives the Apple PhD Fellowship in 2024.

Gang Liu (Graduate Student Member, IEEE) received the BS degree from Southwest University in 2021. He is currently working toward the PhD degree with Computer Science and Engineering, the University of Notre Dame, advised by Prof. Meng Jiang. His research interest is graph machine learning (e.g., prediction and generation) with applications in scientific discovery (e.g., molecules, polymers). He has first-authored publications in top venues like KDD, NeurIPS, and *ACM Transactions on Knowledge Discovery from Data*.



Chi Han is currently working toward the PhD degree in computer science with the University of Illinois at Urbana-Champaign, advised by Prof. Heng Ji. His research interests are centered around understanding of large language models (LLMs) representations to provide insights into and develop useful adaptations for LLMs. He has published first-author papers in NeurIPS, ICLR NAACL and ACL. His work received Outstanding Paper Awards in NAACL 2024 and ACL 2024.

Meng Jiang received the BE and PhD degree from Tsinghua University in 2010 and 2015. He spent two years in UIUC as a postdoc and joined the faculty of University of Notre Dame in 2017, where he is currently an associate professor of Computer Science and Engineering. His research interests include data mining, machine learning, and natural language processing. He was given by NSF the CAREER award in 2022. The honors and awards he received include Best Paper Finalist in KDD 2014, Best Paper Award in KDD-DLG workshop 2020, ACM SIGSOFT Distinguished Paper Award in ICSE 2021, and Outstanding Paper Award in EMNLP 2023.

Heng Ji (Member, IEEE) is a professor with Computer Science Department, and an affiliated faculty member with Electrical and Computer Engineering Department and Coordinated Science Laboratory of University of Illinois Urbana-Champaign. She is an Amazon scholar and the founding director with Amazon-Illinois Center on AI for Interactive Conversational Experiences. Her research interests focus on natural language processing, especially on multimedia multilingual information extraction, knowledge-enhanced large language models, knowledge-driven generation, and conversational AI. She was selected as "Young Scientist" by the World Economic Forum in 2016 and 2017 and was named as part of Women Leaders of Conversational AI (Class of 2023) by Project Voice. The awards she received include "AI's 10 to Watch" Award by IEEE Intelligent Systems in 2013, NSF CAREER award in 2009, PACLIC2012 Best paper runner-up, "Best of ICDM2013" paper award, "Best of SDM2013" paper award, ACL2020 Best Demo Paper Award, NAACL2021 Best Demo Paper Award, Google Research Award in 2009 and 2014, IBM Watson Faculty Award in 2012 and 2014 and Bosch Research Award in 2014-2018.

Jiawei Han (Fellow, IEEE) is Michael Aiken chair professor with the Siebel School of Computing and Data Science, University of Illinois Urbana-Champaign. He has been researching into data mining, text mining, machine learning, and large language models, with more than 1000 publications. He served as the founding editor-in-chief of *ACM Transactions on Knowledge Discovery from Data* (TKDD) (2007-2012). He has received ACM SIGKDD Innovation Award (2004), IEEE Computer Society Technical Achievement Award (2005), IEEE Computer Society W. Wallace McDowell Award (2009), and Japan's Funai Achievement Award (2018). He is fellow of ACM and served as co-director of KnowEnG, a Center of Excellence in Big Data Computing (2014-2019), funded by NIH Big Data to Knowledge (BD2K) Initiative and as the director of Information Network Academic Research Center (INARC) (2009-2016) supported by the Network Science-Collaborative Technology Alliance (NS-CTA) program of U.S. Army Research Lab. His co-authored textbook "Data Mining: Concepts and Techniques" (Morgan Kaufmann) has been adopted popularly as a textbook worldwide.