Bootstrapping Heterogeneous Graph Representation Learning via Large Language Models: A Generalized Approach

Hang Gao[*1], **Chenhao Zhang**[*1, 2], **Fengge Wu**[1, 2†], **Changwen Zheng**[1, 2], **Junsuo Zhao**[1, 2], **Huaping Liu**[3]

[1]National Key Laboratory of Space Integrated Information System, Institute of Software, Chinese Academy of Sciences.
[2]University of Chinese Academy of Sciences.
[3]Tsinghua University.

Abstract

Graph representation learning methods are highly effective in handling complex non-Euclidean data by capturing intricate relationships and features within graph structures. However, traditional methods face challenges when dealing with heterogeneous graphs that contain various types of nodes and edges due to the diverse sources and complex nature of the data. Existing Heterogeneous Graph Neural Networks (HGNNs) have shown promising results but require prior knowledge of node and edge types and unified node feature formats, which limits their applicability. Recent advancements in graph representation learning using Large Language Models (LLMs) offer new solutions by integrating LLMs' data processing capabilities, enabling the alignment of various graph representations. Nevertheless, these methods often overlook heterogeneous graph data and require extensive preprocessing. To address these limitations, we propose a novel method which leverages the strengths of both LLM and GNN, allowing for the processing of graph data with any format and type of nodes and edges without the need for type information or special preprocessing. Our method employs LLM to automatically summarize and classify different data formats and types, aligns node features, and uses a specialized GNN for targeted learning, thus obtaining effective graph representations for downstream tasks. Theoretical analysis and experimental validation have demonstrated the effectiveness of our method.

Code, Datasets and Appendix —
https://github.com/zch65458525/GHGRL/tree/main

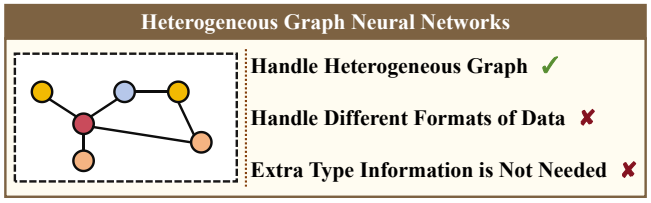
Introduction

Graph representation learning methods are highly effective for processing complex non-Euclidean data, as they can model intricate relationships within graph structures. However, real-world scenarios often involve heterogeneous graph data, which consists of various types of nodes and edges due to the diverse sources and complexity of the data (Wang et al. 2023b). Examples include social network analysis (Qiu et al. 2018; Li and Goldwasser 2019), recommendation systems (Fan et al. 2019b; Yang et al. 2015), and traffic prediction (Guo et al. 2019). General graph representation learning methods often struggle to handle this heterogeneity. Therefore, developing methods that can effectively process and learn from graphs with diverse node and edge types is essential to broaden the applicability of graph representation learning and enhance its capability to manage complex data.

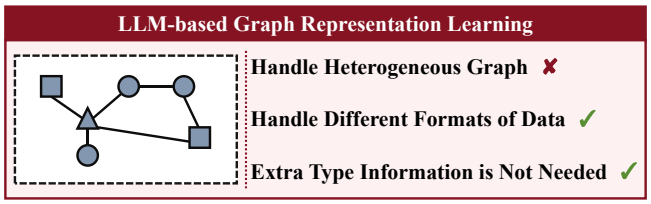
To overcome these difficulties, Heterogeneous Graph Neural Networks (HGNNs) have been developed and shown promising results (Hong et al. 2020; Dong, Chawla, and Swami 2017; Yang et al. 2022). HGNNs are designed to process graphs with varying node and edge types using specialized techniques, including both metapath-based (Wang et al. 2019; Fu et al. 2020) and metapath-free approaches (Fan et al. 2019a). These works leverage meta-path-based aggregation, attention mechanisms, and embedding techniques to effectively manage the diversity of nodes and edges, enabling the processing of heterogeneous graph data. However, HGNNs have limitations that restrict their applicability in scenarios where prior knowledge of node and edge types or consistent node feature formats is unavailable. For example, in open-source intelligence analysis, IoT log analysis, or monitoring malicious internet activities, the unpredictable, diverse, and dynamic nature of the data poses significant challenges in identifying and labeling node types.

Recently, the emergence of graph representation learning methods based on LLMs (Devlin et al. 2019; Brown et al. 2020) has provided new solutions to the aforementioned problems. These methods integrate the background knowledge and data processing capabilities of LLMs into graph representation learning, allowing for the alignment of various types of graph representations based on LLMs (Chen et al. 2023; Huang et al. 2023). These approaches can handle diverse graph data, achieving significant results in the field of graph representation learning and providing directions for building foundational models in this area. However, these methods primarily focus on handling different types of homogeneous graph representation learning tasks and overlook the importance of processing heterogeneous graph data. Nevertheless, an effective method capable of processing such data without the need for additional data cleaning and annotation is highly necessary. At the same time, these methods often require a certain degree of pre-

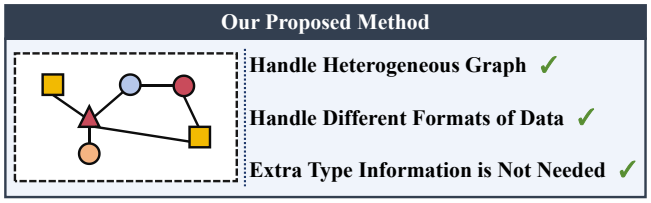
[*]These authors contributed equally.
[†]Corresponding author, fengge@iscas.ac.cn.



(a) The left side of the figure shows the form of input graph data for HGNN, where nodes of different colors represent different types of heterogeneous nodes. The labels for node type and edge type in the graph indicate the required type information for the input data. The right side outlines its characteristics.



(b) Similar to the above figure, the left side of the figure shows the form of input graph data for LLM-based graph representation learning, where nodes of different shapes represent different forms of node attributes.



(c) Demonstration and summary of our method, following the same demonstration format as the two figures above.

Figure 1: Demonstration of different methods.

processing of the graph data, which limits their practical application. Figure 1(a) and 1(b) provide a practical illustration of these methods.

To address these challenges, we propose a novel ***G**eneralized **H**eterogeneous **G**raph **R**epresentation **L**earning* (GHGRL) method. GHGRL integrates the strengths of both LLMs and GNNs to process graph data in a more generalized manner. As demonstrated in Figure 1(c), GHGRL can handle graph data with nodes and edges of any format and type, without requiring explicit type information or special pre-processing of the data. Specifically, GHGRL utilizes LLMs to process the training data by automatically summarizing and classifying the various data formats and types present in the graph. Subsequently, LLMs are used to align node features across different formats, generating representation vectors for node attributes. Next, we employ our specially designed GNN to perform targeted learning on the graph data based on the types and estimations derived from the LLM, thereby obtaining graph representations suitable for downstream tasks.

Contributions:

- We propose a novel method that combines LLM and GNN to process heterogeneous graph data without requiring node and edge type information. Additionally, this method can handle scenarios where node attributes are not uniform.
- We present the specific implementation of the aforementioned method and conduct theoretical analysis and validation of its performance.
- We developed more challenging datasets to rigorously test the proposed method. Additionally, we validated our approach using widely adopted heterogeneous graph datasets to ensure robustness and reliability.

Related Works

Heterogeneous Graph Representation Learning. Heterogeneous graph representation learning methods are categorized into metapath-based and metapath-free approaches. Metapath-based methods use heterogeneous graph neural networks to aggregate and integrate semantic features (Yun et al. 2019; Zhang et al. 2019; Wang et al. 2019; Fu et al. 2020; Bing et al. 2023; Yang et al. 2023). HetGNN (Zhang et al. 2019) uses random walks and node type aggregation. HAN (Wang et al. 2019) and MAGNN (Fu et al. 2020) use metapaths for semantic differentiation and propagation. SeHGNN (Yang et al. 2023) extends receptive fields with long metapaths and transformer-based modules. Metapath-free methods embed semantic information using attention mechanisms (Zhu et al. 2019; Fan et al. 2019a; Hong et al. 2020; Lv et al. 2021; Zhou et al. 2023; He et al. 2024a). HGB (Lv et al. 2021) uses a multi-layer GAT network for node distinction. PSHGCN (He et al. 2024a) uses positive spectral heterogeneous graph convolution to learn valid heterogeneous graph filters. These methods all require prior knowledge of node and edge types and are typically used on datasets where these types are known. However, this limitation restricts the application of these methods in the broader field of data mining.

LLMs for Graphs. With the emergence of various LLM methods, the use of LLMs for graph representation learning is becoming a research hotspot. Relevant studies can be classified into two types. One type enriches node representation based on prompt learning and processes graph data tasks using GNN (Fatemi, Halcrow, and Perozzi 2023; Chen et al. 2023; Huang et al. 2023, 2024; Liu et al. 2024; Tang et al. 2024). The other type converts graph data into text for LLM processing (Ai et al. 2023; Wang et al. 2023a; Guo, Du, and Liu 2023; Sun et al. 2023; Luo et al. 2024; Tan et al. 2024). These methods handle homogeneous data and require pre-standardized representation of node information, limiting their application in data mining. To address this, our proposed method is designed to integrate LLMs to handle heterogeneous graph attributes of any format and type without prior knowledge, expanding the application scope in data mining. Please refer to **Appendix** A for an extended related work.

Methodology

Our proposed GHGRL framework aims to enhance learning on heterogeneous graphs using LLMs, thereby making the processing capabilities more generalized. Specifically,

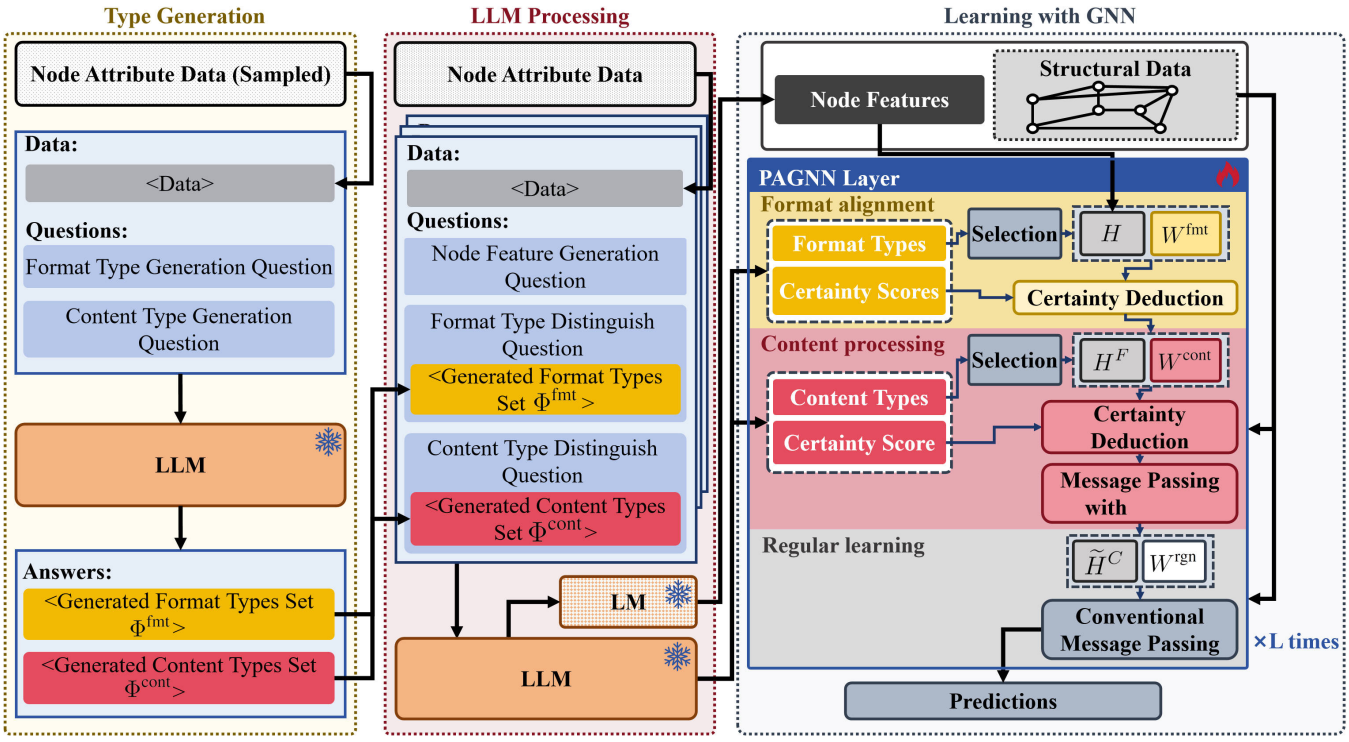


Figure 2: The framework of the proposed method. The snowflake symbol represents the fixed model parameters, while the flame represents the model parameters involved in training.

for a heterogeneous graph $G = \{\mathcal{V}, \mathcal{E}\}$, G contains various types of nodes with different representation formats. Additionally, the edges between adjacent nodes may also possess different types. Moreover, the types of nodes and edges are unknown to us. Our goal is to construct a model that can effectively handle G and accomplish graph representation learning tasks. To achieve this, GHGRL is divided into the following three modules: **1) Type Generation**, which identifies all possible node types based on node attributes; **2) LLM Processing**, which estimates the specific type of each node and generates node representation vectors using LLMs; and **3) Learning with GNN**, which leverages the acquired node types and representations for graph learning with a specially designed GNN. During the GNN learning process, different types of edges are distinguished, and message passing is executed accordingly. Figure 2 demonstrates the framework of GHGRL.

Type Generation

First, since we do not have access to the number of node types in the dataset or detailed information about them, we opt to generate these types directly. We create two categories of node type set: the format-based set Φ^{fmt} and the content-based set Φ^{cont}. Specifically, we randomly select a subset of node attribute samples, denoted as $\tilde{\boldsymbol{X}} = \{\boldsymbol{x}_i\}_{i=1}^{|\widetilde{X}|}$, from the training set and input them into the LLM, allowing it to analyze and identify the node types present in the dataset. We use Llama 3 (Dubey et al. 2024) as our backbone LLM. The number of selected samples, $|\widetilde{X}|$, is determined by the maximum input sequence length of the LLM.

As a result, we obtain the generated format types set $\Phi^{\text{fmt}} = \{s_j^{\text{fmt}}\}_{j=1}^{m^{\text{fmt}}}$, where each s_j^{fmt} represents a generated string-formatted node format type name, represented in text format, such as "noun" or "detail description". Similarly, the generated content types set is $\Phi^{\text{cont}} = \{s_j^{\text{cont}}\}_{j=1}^{m^{\text{cont}}}$, where each s_j^{cont} denotes a generated string-formatted node content type name, also represented in text format, such as "paper concerning deep learning" or "paper concerning biology". The parameters m^{fmt} and m^{cont} are hyperparameters controlling the size of Φ^{fmt} and Φ^{cont}, respectively. Formally, we have:

$$\left\{\Phi^{\text{fmt}}, \Phi^{\text{cont}}\right\} = LLM\left(\tilde{\boldsymbol{X}}, m^{\text{fmt}}, m^{\text{cont}}, P^G\right), \quad (1)$$

where P^G denotes the type generation prompt. Please refer to **Appendix** C for details.

LLM Processing

Subsequently, we process the data with the LLM to acquire node features, estimating the format type and content type of each node's attribute features. Based on Φ^{fmt} and Φ^{cont}, we conduct analysis upon each node v's feature $\boldsymbol{x}_v$. We obtain five different outputs: description text $\boldsymbol{h}_v^{\text{desc}}$ of node v, format type estimation result $\phi^{\text{fmt}}(v)$, format type estimation confidence score $c^{\text{fmt}}(v)$, content type estimation result $\phi^{\text{cont}}(v)$, content type estimation confidence score $c^{\text{cont}}(v)$, description text $\boldsymbol{h}_v^{\text{reas}}$ of the estimation reasons. $\phi^{\text{fmt}}(v)$ denotes the index of the estimated format type of node v within Φ^{fmt}, while $\phi^{\text{cont}}(v)$ denotes the index of the estimated con-

tent type of node v within Φ^{cont}. Formally, we have:

$$\left\{\boldsymbol{h}_v^{\text{desc}}, \phi^{\text{fmt}}(v), c^{\text{fmt}}(v), \phi^{\text{cont}}(v), c^{\text{cont}}(v), \boldsymbol{h}_v^{\text{reas}}\right\} = LLM\left(\boldsymbol{x}_v, \Phi^{\text{fmt}}, \Phi^{\text{cont}}, P^P\right). \quad (2)$$

We ensure that the LLM outputs as much information about the node attributes as possible by modifying the prompts. Please refer to **Appendix** C for details. In addition to node descriptions, we also require the model to provide a reasoning description for its estimation of the node types. This approach allows the model to refine and optimize the modeling of heterogeneous graphs.

Subsequently, we adopt a language model sentence transformer (Reimers and Gurevych 2019) to generate fixed-length node representation based on both $\boldsymbol{h}_v^{\text{desc}}$ and $\boldsymbol{h}_v^{\text{reas}}$, formally, we have:

$$\boldsymbol{h}_v = s(\boldsymbol{h}_v^{\text{desc}}, \boldsymbol{h}_v^{\text{reas}}), \quad (3)$$

where $s(\cdot)$ denotes the sentence transformer model. $\boldsymbol{h}_v$ will be utilized as the node feature in the subsequent modules.

Learning with GNN

To integrate LLM estimates into graph representation learning, we specifically designed a novel ***P**arameter **A**daptive **GNN*** (PAGNN) to maximize the utilization of LLM outputs for graph data processing. Part of the PAGNN structure is determined by the LLM outputs. Specifically, each layer of PAGNN includes three components: a format alignment block based on format type, a heterogeneous processing block based on content type, and a regular learning block. These components will be introduced in detail below.

Format alignment block. The purpose of this block is to align node features represented in different forms. This block utilizes matrix $W^{\text{fmt}} \in \mathbb{R}^{|\Phi^{\text{fmt}}| \times d^{\text{in}} \times d^{\text{fmt}}}$ and $B^{\text{fmt}} \in \mathbb{R}^{|\Phi^{\text{fmt}}| \times d^{\text{fmt}}}$ as the network parameters, where d^{in} and d^{fmt} denote the input and output feature width of this block, $|\Phi^{\text{fmt}}|$ is the number of format types. Subsequently, with the input node representation matrix H, where $H \in \mathbb{R}^{|\mathcal{V}| \times x^{\text{fmt}}}$, $|\mathcal{V}|$ is the number of nodes, for all $v \in \{1, 2, ..., |\mathcal{V}|\}$ the block performs the following calculation:

$$H^{\text{fmt }[v]} = \delta\left(H^{[v]} W^{\text{fmt }[\phi^{\text{fmt}}(v)]} + B^{\text{fmt }[\phi^{\text{fmt}}(v)]}\right), \quad (4)$$

where $W^{\text{fmt }[\phi^{\text{fmt}}(v)]}$ and $B^{\text{fmt }[\phi^{\text{fmt}}(v)]}$ denote the $\phi^{\text{fmt}}(v)$-th elements of W^{fmt} and B^{fmt} along the first dimension, respectively. $H^{\text{fmt }[v]}$ denotes the v-th vector within H^{fmt}. This design ensures that all nodes with the same type utilize the same set of parameters, while nodes with different types utilize different sets of parameters. Due to the potential inaccuracy and possible misjudgment of node type estimation by LLM, we further introduce the generated confidence score $c^{\text{fmt}}(v)$ and adjust the blocks based on this score. We optimize Equation 4 to generate a formal representation of the block with the added confidence score as follows:

$$H^{\text{fmt }[v]} = \delta\Big(c^{\text{fmt}}(v)\left(H^{[v]} W^{\text{fmt }[\phi^{\text{fmt}}(v)]} + B^{\text{fmt }[\phi^{\text{fmt}}(v)]}\right) + \left(1 - c^{\text{fmt}}(v)\right) H^{[v]}\Big), \quad (5)$$

where $c^{\text{fmt}}(v)$ is the aforementioned format type confidence score of v. $\delta(\cdot)$ denotes the activate function. This design ensures that the effect of $W^{\text{fmt }[\phi^{\text{fmt}}(v)]}$ decreases as the confidence score decreases.

Content processing block. This block trails the format alignment block. It processes node features of different generated node content types and then conducts message passing between them. The pattern of information transmission between different nodes may vary. Conventional heterogeneous graph representation learning methods often use meta paths or predefined edge types to address this issue (Wang et al. 2019; Fu et al. 2020; Yang et al. 2023). However, since we cannot obtain this information, we can only differentiate information transmission based on the node content type generated by the LLM. Specifically, the content processing block first conducts the following calculation:

$$H^{\text{cont }[v]} = \delta\Big(c^{\text{cont}}(v)\big(H^{\text{fmt }[v]} W^{\text{cont }[\phi^{\text{cont}}(v)]} + B^{\text{cont }[\phi^{\text{cont}}(v)]}\big)\Big), \quad (6)$$

where $c^{\text{cont}}(v)$ is the aforementioned content type confidence score of v, $W^{\text{cont}} \in \mathbb{R}^{|\Phi^{\text{cont}}| \times d^{\text{fmt}} \times d^{\text{cont}}}$ and $B^{\text{cont}} \in \mathbb{R}^{|\Phi^{\text{cont}}| \times d^{\text{cont}}}$ are parameter matrices. d^{cont} denotes the feature width of each representation vector within $H^{\text{cont }[v]}$. The content processing block then conducts message passing with H^{cont}:

$$\widetilde{H}^{\text{cont }[v]} = \alpha H^{\text{cont }[v]} + AGG\left(H^{\text{cont }[u]} \widetilde{W}^{\text{cont }[\phi(v)]}, u \in \mathcal{N}(v)\right), \quad (7)$$

where α be a hyperparameter to control the proportion of original node features, parameter matrix $\widetilde{W}^{\text{cont}} \in \mathbb{R}^{|\Phi^{\text{cont}}| \times d^{\text{cont}} \times d^{\text{cont}}}$. $AGG(\cdot)$ aggregates the features from neighbors. We adopt d^{cont} again as the output feature width of content processing block. Equation 6 and 7 actually ensure that during the aggregation operation, the node representations are multiplied by the corresponding parameter matrices according to the content type of the source and target nodes of the edges. This, in turn, ensures that the entire data aggregation process maximally distinguishes between different node types and edge types.

Regular learning block. This block follows the first two blocks and can be formally represented as follows.

$$H^{\text{rgn }[v]} = \delta\Big(\widetilde{H}^{\text{cont }[v]} W^{\text{rgn}} + AGG\left(\widetilde{H}^{\text{cont }[u]} W^{\text{rgn}}, u \in \mathcal{N}(v)\right)\Big), \quad (8)$$

which is similar to a regular GCN layer, adopting the same method for data propagation to learn the common features present in the data. $W^{\text{rgn}} \in \mathbb{R}^{d^{\text{cont}} \times d^{\text{rgn}}}$ is the parameter matrix. d^{rgn} denotes the output feature width of this block.

The aforementioned blocks form a PAGNN layer. Our model is composed of multiple PAGNN layers, and we remove the format alignment block and the content processing block after the l^{fmt} layer and l^{cont} layer respectively, as the

heterogeneity of node features is sufficiently represented by that point. $l^{\text{fmt}} \leq l^{\text{cont}} \leq L$, where L is the total number of network layers. Please refer to **Appendix** D for details.

Analysis

In this section, we further analyze the proposed method by examining how GHGRL effectively learns various types of semantics. This capability helps to mitigate semantic confusion, which can arise from the over-smoothing common in conventional graph representation learning methods. Such over-smoothing can be problematic when dealing with complex heterogeneous graph data (Zhou et al. 2023). We adopt a simplified graph convolution model $g(\cdot)$ (Kipf and Welling 2017) for this type of analysis, the layer structure of which can be represented as follows:

$$\boldsymbol{h}_v^{(l+1)} = \boldsymbol{h}_v^{(l)} + AGG\left(\boldsymbol{h}_u^{(l)} W^{(l)}, u \in \mathcal{N}(v)\right), \quad (9)$$

where $\boldsymbol{h}_v^{(l+1)}$ and $\boldsymbol{h}_v^{(l)}$ denote the output representation of node v of layer l and $l+1$ respectively. Several related works (Zhou et al. 2023; Li, Han, and Wu 2018) have demonstrated that this model can effectively represent the properties of different types of graph neural networks, making it widely applicable in the analysis of the over-smoothing characteristics of graph neural networks. Furthermore, it bears significant similarity to our model's architecture.

GNN models generally suffer from over-smoothing (Chen et al. 2020; Geerts and Reutter 2022). Specifically, for graph convolution model $g(\cdot)$ with L layers and graph $G = \{\mathcal{V}, \mathcal{E}\}$, we can represent the limitation of g when $L \to +\infty$:

$$\lim_{L\to+\infty} g(\mathcal{V}, \mathcal{E}) = \begin{bmatrix} \boldsymbol{h}_1^{(L)} & \boldsymbol{h}_2^{(L)} & \cdots & \boldsymbol{h}_{|\mathcal{V}|}^{(L)} \end{bmatrix}^\top, \quad (10)$$

where $\boldsymbol{h}_v^{(L)}$ denotes the L-th layer output representation of node v. For any node i and j within G, $\boldsymbol{h}_i^{(L)}$ and $\boldsymbol{h}_j^{(L)}$ are linearly dependent. Yet, our proposed GHGRL could avoid such over-smoothing. To prove this, we construct a simplified model $\widetilde{g}(\cdot)$ of GHGRL, l-th layer of $\widetilde{g}(\cdot)$ possesses the following form:

$$\boldsymbol{h}_v^{(l+1)} = \boldsymbol{h}_v^{(l)} + AGG\left(\boldsymbol{h}_u^{(l)} W^{[\phi(v)]} + B^{[\phi(v)]}, u \in \mathcal{N}(v)\right). \quad (11)$$

The input of $\widetilde{g}(\cdot)$ are node features extracted from LLM $f(\cdot)$. Subsequently, we propose the following theorem.

Theorem 1. *Given a connected graph $G = \{\mathcal{V}, \mathcal{E}\}$ with node features $\{\boldsymbol{x}_i\}_{i=1}^{|\mathcal{V}|}$ and LLM $f(\cdot)$, $\widetilde{g}(\cdot)$ can avoid the over-smoothing described in Equation 10 for the node features, i.e., we have:*

$$\lim_{L\to+\infty} \widetilde{g}(\{f(\boldsymbol{x}_j)\}_{j=1}^{|\mathcal{V}|}, \mathcal{E}) = \begin{bmatrix} \tilde{\boldsymbol{h}}_1^{(L)} & \tilde{\boldsymbol{h}}_2^{(L)} & \cdots & \tilde{\boldsymbol{h}}_{|\mathcal{V}|}^{(L)} \end{bmatrix}^\top, \quad (12)$$

where for node i and j that satisfying $\phi(i) \neq \phi(j)$, $\tilde{\boldsymbol{h}}_i$ and $\tilde{\boldsymbol{h}}_j$ are linearly independent.

The proof can be found in **Appendix** B.1. Theorem 1 demonstrates through the model in Equation 11 that our method effectively prevents over-smoothing among different types of nodes. This ensures that the model preserves the distinctive features between various node types. Moreover, this differentiation is automatically derived based on the judgments produced by an LLM, ensuring that our model can leverage the knowledge of the LLM for type estimation. Consequently, this facilitates relation learning training based on the structure of the GHGRL.

Corollary 2. *Given the conditions in Theorem 1, if node i and j satisfied $\phi(i) = \phi(j)$, i and j do not share same set of neighbors, then $\tilde{\boldsymbol{h}}_i^{(L)}$ and $\tilde{\boldsymbol{h}}_j^{(L)}$ are not necessarily linear dependent for $L \to +\infty$.*

The proof can be found in **Appendix** B.2. Corollary 2 further demonstrates that our proposed method not only prevents over-smoothing between different types of heterogeneous nodes, but also ensures that over-smoothing does not necessarily occur between nodes of the same type. In such cases, whether over-smoothing occurs depends on the types of adjacent nodes and network parameters. This means the network can adaptively make node features similar or different based on specific circumstances, rather than causing all node features to converge to the same value due to over-smoothing. As shown in (Li, Han, and Wu 2018), a 3-layer GCN can already experience over-smoothing on certain datasets. With two aggregations per layer, our 3-layer PAGNN is equivalent to a 6-layer GCN (Please refer to Section Experiments for details), heightening the risk of over-smoothing. We analyzed inter-type node similarity across layers and compared our method to GHGRL without PAGNN (GHGRL w/o P). The results below confirm that over-smoothing occurs, and our method effectively prevents it.

Method	Layer 1	Layer 2	Layer 3	Layer 4
GHGRL w/o P	-0.151	0.327	0.702	0.882
GHGRL	-0.133	0.174	0.311	0.395

Table 1: Mean cosine similarity between each type's average feature vector and the overall mean, indicating the degree of over-smoothing on the IMDB dataset.

Experiments

Comparison with State of the Art methods

Baselines. For baseline methods, we compared our approach with three categories of baselines: 1) general GNN backbone networks, including GCN (Kipf and Welling 2017) and GAT (Velickovic et al. 2018), 2) HGNN methods, including HAN (Wang et al. 2019), MAGNN (Fu et al. 2020), SeHGNN (Yang et al. 2023) and PSHGCN (He et al. 2024a) and 3) more generalized graph representation learning methods that combines GNN and LLM, including TAPE (He et al. 2024b), OFA (Liu et al. 2024), and GOFA (Kong et al. 2024).

Datasets. We utilized existing commonly used heterogeneous and homogeneous graph representation learning datasets, as well as more challenging heterogeneous graph

Datasets	IMDB (10% Training)		IMDB (40% Training)		DBLP (10% Training)		DBLP (40% Training)	
Metrics	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
GCN	57.47±0.72	58.43 ±1.15	60.13±0.76	60.38±1.19	89.09±0.32	89.8±0.34	88.94±0.38	89.61±0.40
GAT	60.12±0.79	60.79±1.26	62.85±1.28	63.1±0.83	89.66±0.26	90.93±0.23	91.40±0.19	91.79±0.21
HAN	61.28±0.12	61.26±0.15	62.78±0.38	62.15±0.26	91.23±0.51	92.10±0.62	91.92±0.48	92.52±0.59
MAGNN	57.78±2.85	57.97±1.82	59.92±1.24	60.07±0.89	92.24±0.49	92.70±0.51	93.21±0.42	93.68±0.43
SeHGNN	61.23±0.46	62.74±0.37	62.62±0.35	65.34±0.30	**93.74±0.28**	**94.19±0.24**	**94.48±0.12**	**94.85±0.15**
PSHGCN	61.35±0.79	62.25±0.42	67.21±0.66	67.55±0.56	92.89±0.09	93.46±0.07	93.98±0.12	94.29±0.10
HAN-w	58.31±0.32	58.26±0.31	59.83±0.33	59.02±0.35	87.54±0.78	87.93±0.58	88.16±0.68	88.64±0.69
MAGNN-w	57.02±1.23	57.36±0.86	59.44±1.06	59.76±0.93	90.24±0.49	90.65±0.63	90.32±0.77	91.32±0.82
SHEGNN-w	59.56±0.78	61.30±1.34	61.76±0.62	65.24±0.73	89.32±0.28	89.96±0.24	91.57±0.21	91.71±0.22
PSHGCN-w	59.68±0.55	61.04±0.36	65.52±0.52	66.03±0.44	89.78±0.23	90.46±0.25	91.58±0.12	91.93±0.10

Table 2: Comparative experiment results for IMDB and DBLP datasets. **Bold** denotes the best performance, underline denotes the second best. "-w" denotes results of HGNN method without type information.

Datasets	ACM (10% Training)		ACM (40% Training)	
Metrics	Macro-F1	Micro-F1	Macro-F1	Micro-F1
GCN	89.47±0.23	90.23±0.24	89.19±0.28	89.95±0.27
GAT	92.23±0.96	92.27±0.95	92.26±0.86	92.38±0.81
HAN	90.58±0.40	90.56±0.39	92.70±0.45	92.75±0.42
MAGNN	89.46±0.64	89.71±0.53	91.25±0.24	91.33±0.35
SeHGNN	92.06±0.32	92.10±0.32	93.38±0.30	93.44±0.36
PSHGCN	91.07±0.26	91.00±0.24	93.78±0.23	93.77±0.19
HAN-w	90.08±0.34	90.02±0.32	91.98±0.38	91.86±0.37
MAGNN-w	88.95±0.18	89.26±0.20	91.23±0.16	91.33±0.25
SHEGNN-w	91.87±0.48	91.81±0.36	92.45±0.42	92.50±0.44
PSHGCN-w	91.01±0.26	90.97±0.24	92.97±0.23	93.02±0.19
TAPE	78.26±0.95	78.63±0.97	88.91±0.76	88.81±0.62
OFA	72.63±0.23	72.34±0.16	80.32±0.24	80.65±0.28
GOFA	78.91±0.56	78.92±0.73	84.28±0.33	84.21±0.79
GHGRL	**92.71±0.36**	**92.69±0.30**	**94.21±0.44**	**94.63±0.42**

Table 3: Comparative experiment results for ACM datasets. **Bold** denotes the best performance, underline denotes the second best. "-w" denotes results of HGNN method without type information.

datasets that we newly constructed. Specifically, we employed the IMDB, DBLP, ACM (Zhang et al. 2019) and Wiki-CS (Mernyei and Cangea 2020) datasets , and we reported the test accuracy under varying amounts of training data. Additionally, we constructed two new datasets, the ***R**andom **I**nformation **R**eplacement on **IMDB*** (IMDB-RIR) and the ***R**andom **I**nformation **D**eletion on **DBLP*** (DBLP-RID): We utilized both commonly used heterogeneous graph representation learning datasets and more challenging datasets that we newly constructed. Specifically, we employed the IMDB, DBLP, and ACM datasets (Zhang et al. 2019), and reported the test accuracy with varying amounts of training data. Additionally, we constructed two new datasets: ***IMDB** dataset with **R**andom **I**nformation **R**eplacement* (IMDB-RIR) and ***DBLP** dataset with **R**andom **I**nformation **D**eletion* (DBLP-RID).

- **IMDB-RIR.** Based on the IMDB dataset, we performed searches on Google using the textual information of the nodes in the IMDB dataset. We then saved the top 10 search results for each node. Subsequently, we randomly selected results from these top 10 and used them to replace the node attributes in the IMDB dataset. As a result, the constructed dataset contains information in various uncertain formats, thereby increasing the complexity of the tasks.
- **DBLP-RID.** Based on the DBLP dataset, we randomly deleted portions of the node textual information, creating a new graph dataset with partially missing node information.

These datasets introduce further diversity into heterogeneous datasets and are utilized for extra comparison. Further details can be found in **Appendix** D.1.

Settings. We followed the basic settings outlined in OFA (Liu et al. 2024) and used Llama 3 (Dubey et al. 2024) as the LLM for both our method and the baseline methods to ensure a fair comparison. Additionally, we adjusted the proportion of training data in the datasets to compare test results under different conditions. For all experimental results, we conducted five independent runs and reported the mean ± standard deviation. The specific experimental setup, including hyperparameters and the environment used, is detailed in **Appendix** D.

Results on heterogeneous graph datasets. Table 2 and 3 demonstrate the results of experiments on IMDB, DBLP, and ACM. Since our approach does not use the node type or edge type information included in the heterogeneous graph datasets as input, for better analysis, we compared our method with HGNN baselines that both use and do not use this information. We mark the methods that do not utilize this information with "-w". In the results, we can see that our method achieves either the best performance or performance comparable to methods that use additional type information on all datasets, demonstrating the capability of GHGRL.

Results on heterogeneous graph datasets with extra diversity. Table 4 demonstrates the results of experiments on IMDB-RIR and DBLP-RID. We denote r as the proportion of newly constructed data used in the dataset, e.g., 20% denotes we utilize 20% of the total amount of newly constructed data. Since within IMDB-RIR and DBLP-RID, the node features have been modified by the additional information we introduced, they no longer adhere to a standard format. Consequently, GNN and HGNN-based methods can no

Datasets	IMDB-RIR (r=20%)		IMDB-RIR (r=100%)		DBLP-RID (r=20%)		DBLP-RID (r=100%)	
Metrics	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
Llama3	40.25±0.95	39.35±0.90	39.51±0.81	39.77±0.51	36.56±1.15	46.16±0.95	43.51±1.39	43.73±0.56
TAPE	48.59±1.10	48.71±0.61	40.88±0.74	40.57±0.62	54.87±0.84	54.71±1.11	50.24±0.51	50.65±0.66
OFA	20.44±0.12	20.78 ±0.26	21.13±0.12	21.21±0.12	31.35±0.08	31.83±0.17	30.52±0.25	30.23±0.27
GOFA	33.18±0.62	33.91±0.81	29.16±0.79	29.09±0.67	40.75±0.58	40.28±0.89	37.11±0.87	37.32±0.78
GHGRL	**75.15±0.43**	**75.35 ±0.77**	**74.53±0.56**	**74.83±0.52**	**93.47±0.21**	**93.72±0.25**	**91.20±0.78**	**91.83±0.92**

Table 4: Comparative experiment results for heterogeneous graph datasets with extra diversity. **Bold** denotes the best performance, underline denotes the second best.

Datasets	IMDB-RIR (r=20%)		IMDB-RIR (r=60%)		IMDB-RIR (r=100%)	
Metrics	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
GCN + LLM	64.26±0.14	64.16±0.29	59.18±0.63	60.98±0.71	58.41±0.35	58.59±0.71
GAT + LLM	65.28±0.60	65.30±0.70	65.84±0.76	65.77±0.81	65.25±0.57	65.42±0.28
HAN + LLM	64.89±0.83	65.07±0.23	64.59±0.63	64.60±0.55	63.81±0.46	63.90±0.30
MAGNN + LLM	61.88±0.56	61.92±0.47	61.42±0.46	61.36±0.87	61.34±0.19	61.32±0.52
SeHGNN + LLM	68.35±0.50	68.82±0.34	68.21±0.81	68.62±0.64	67.76±0.44	68.06±0.33
PSHGCN + LLM	71.83±0.23	72.18±0.47	72.36±0.78	72.77±0.91	72.28±0.60	72.65±0.14
HAN-w + LLM	63.41±0.90	63.52±0.38	63.29±0.89	63.30±0.08	63.88±0.63	63.01±0.24
MAGNN-w + LLM	61.84±0.32	61.87±0.16	60.58±0.51	60.71±0.86	61.24±0.11	61.33±0.68
SeHGNN-w + LLM	66.24±0.57	66.33±0.37	66.02±0.26	66.09±0.35	65.79±0.39	65.92±0.71
PSHGCN-w + LLM	71.43±0.32	71.53±1.12	70.89±0.38	71.07±0.21	70.21±1.06	70.43±0.83
GHGRL	**75.15±0.43**	**75.35±0.77**	**74.72±0.45**	**75.00±0.42**	**74.93±0.46**	**75.15±0.51**

Table 5: Comparative experiment results for HGNNs attached with LLM modules (marked with "+ LLM"). **Bold** denotes the best performance, underline denotes the second best.

Dataset	Wiki-CS	
Metrics	Macro-F1	Micro-F1
GCN	69.78±0.53	75.10±0.58
GAT	70.88±0.50	78.04±0.63
TAPE	77.30±0.59	77.24±0.67
OFA	77.69±0.12	78.32±0.15
GOFA	78.65±0.68	78.74±0.95
GHGRL	**80.69±0.60**	**81.39±0.27**

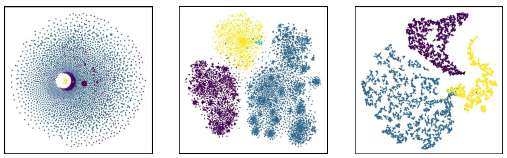
Table 6: Comparative experimental results for the homogeneous dataset. **Bold** indicates the best performance, while underline indicates the second best.

longer process this information without additional help, so we did not include comparisons with these methods. Additionally, we used our LLM, Llama 3, to directly classify the nodes. Here, we can see that while LLM-based methods can somewhat handle our newly constructed dataset, GHGRL still achieved the best performance, significantly surpassing other baseline methods and demonstrating its capability.

Furthermore, in Table **??**, we also integrated the LLM processing module we used into other HGNN methods to output unified features for further comparison on the IMDB-RIR dataset. As shown in the results, even under these conditions, GHGRL still achieved the best performance, significantly outperforming other methods. This demonstrates the strong compatibility between our PAGNN and the LLM module.

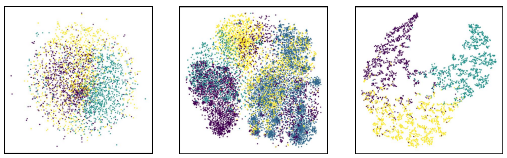
Results on homogeneous graph dataset. We also conducted method comparisons on homogeneous graph datasets. The results within Table 6 show that GHGRL can achieve better performance on homogeneous graphs as well, indicating that its mechanism positively enhances graph representation learning even on standard homogeneous graph data. Further experimental results can be found in **Appendix** E.

In-Depth Analysis



(a) Input. (b) LLM Processed. (c) Output.

Figure 3: Data representations at different stages of the model after dimensionality reduction using the t-SNE method. Different colors represent distinct types of nodes.



(a) Input. (b) LLM Processed. (c) Output.

Figure 4: Data representations at different stages of the model after dimensionality reduction using the t-SNE method. Different colors represent distinct classes of nodes.

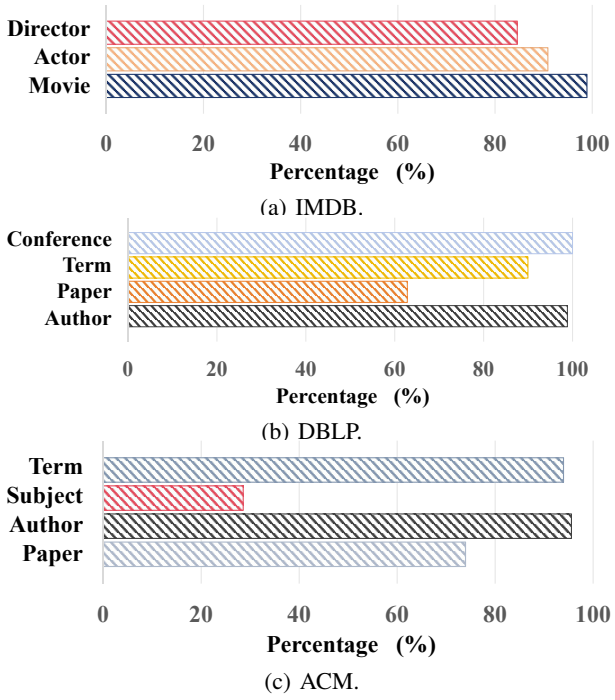


Figure 5: Demonstration of different methods.

Feature visualization. We visualized the node features of the model at different stages on the ACM dataset using the t-SNE method, as shown in Figures 3 and 4. From these figures, it is evident that at the input stage, the node features are highly mixed. However, after being processed by the LLM, these features display multiple dispersed clusters. This indicates that the LLM has leveraged its knowledge to perform a more detailed grouping of the samples. However, this grouping does not align with the desired three-class categorization of the nodes, as some node features remain intermixed. Finally, after processing by PAGNN, our model successfully categorizes the nodes into three distinct groups according to their classes, demonstrating that PAGNN has further refined the information extracted from the LLM's output, ultimately leading to a more optimal result.

LLM processing analysis. Here, we report the statistics on the match between the node types estimated by the model and the actual node types in the IMDB, DBLP, and ACM datasets. The proportion of correctly classified types for each category is summarized in Figure 5. It is evident that our model does not accurately estimate all types. This reveals an interesting phenomenon: our LLM Processing module classifies nodes in the dataset based on its own internal knowledge. Moreover, the results of the aforementioned comparative experiments demonstrate that our model outperforms other models, indicating that GHGRL effectively leverages PAGNN to adapt to the estimations made by the LLM Processing module. As a result, even when there is a discrepancy between the classification and the actual dataset, our model can still achieve satisfactory performance.

Conclusion

In this paper, we propose an innovative approach called GHGRL, which integrates LLM and GNN using an adaptive parameter selection method. This approach enhances the generalization capability for handling heterogeneous graph data, offering a new perspective for processing more complex and irregularly structured graph data.

Acknowledgments

We would like to express our sincere gratitude to the reviewers of this paper, as well as the Program Committee and Area Chairs, for their valuable comments and suggestions. This work is supported by the CAS Project for Young Scientists in Basic Research, Grant No. YSBR-040.

References

Ai, Q.; Zhou, J.; Jiang, H.; Liu, L.; and Shi, S. 2023. When Graph Data Meets Multimodal: A New Paradigm for Graph Understanding and Reasoning. *CoRR*, abs/2312.10372.

Bing, R.; Yuan, G.; Zhu, M.; Meng, F.; Ma, H.; and Qiao, S. 2023. Heterogeneous graph neural networks analysis: a survey of techniques, evaluations and applications. *Artif. Intell. Rev.*, 56(8): 8003–8042.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. *CoRR*, abs/2005.14165.

Chen, M.; Wei, Z.; Huang, Z.; Ding, B.; and Li, Y. 2020. Simple and Deep Graph Convolutional Networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, 1725–1735. PMLR.

Chen, Z.; Mao, H.; Li, H.; Jin, W.; Wen, H.; Wei, X.; Wang, S.; Yin, D.; Fan, W.; Liu, H.; and Tang, J. 2023. Exploring the Potential of Large Language Models (LLMs)in Learning on Graphs. *SIGKDD Explor.*, 25(2): 42–61.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Burstein, J.; Doran, C.; and Solorio, T., eds., *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, 4171–4186. Association for Computational Linguistics.

Dong, Y.; Chawla, N. V.; and Swami, A. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, 135–144. ACM.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; Goyal, A.; Hartshorn, A.; Yang, A.; Mitra, A.; and et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783.

Fan, S.; Zhu, J.; Han, X.; Shi, C.; Hu, L.; Ma, B.; and Li, Y. 2019a. Metapath-guided Heterogeneous Graph Neural Network for Intent Recommendation. In Teredesai, A.; Kumar, V.; Li, Y.; Rosales, R.; Terzi, E.; and Karypis, G., eds., *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, 2478–2486. ACM.

Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, Y. E.; Tang, J.; and Yin, D. 2019b. Graph Neural Networks for Social Recommendation. In Liu, L.; White, R. W.; Mantrach, A.; Silvestri, F.; McAuley, J. J.; Baeza-Yates, R.; and Zia, L., eds., *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, 417–426. ACM.

Fatemi, B.; Halcrow, J.; and Perozzi, B. 2023. Talk like a Graph: Encoding Graphs for Large Language Models. *CoRR*, abs/2310.04560.

Fu, X.; Zhang, J.; Meng, Z.; and King, I. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In Huang, Y.; King, I.; Liu, T.; and van Steen, M., eds., *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, 2331–2341. ACM / IW3C2.

Geerts, F.; and Reutter, J. L. 2022. Expressiveness and Approximation Properties of Graph Neural Networks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Guo, J.; Du, L.; and Liu, H. 2023. GPT4Graph: Can Large Language Models Understand Graph Structured Data ? An Empirical Evaluation and Benchmarking. *CoRR*, abs/2305.15066.

Guo, S.; Lin, Y.; Feng, N.; Song, C.; and Wan, H. 2019. Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, 922–929. AAAI Press.

He, M.; Wei, Z.; Feng, S.; Huang, Z.; Li, W.; Sun, Y.; and Yu, D. 2024a. Spectral Heterogeneous Graph Convolutions via Positive Noncommutative Polynomials. In Chua, T.; Ngo, C.; Kumar, R.; Lauw, H. W.; and Lee, R. K., eds., *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, 685–696. ACM.

He, X.; Bresson, X.; Laurent, T.; Perold, A.; LeCun, Y.; and Hooi, B. 2024b. Harnessing Explanations: LLM-to-LM Interpreter for Enhanced Text-Attributed Graph Representation Learning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Hong, H.; Guo, H.; Lin, Y.; Yang, X.; Li, Z.; and Ye, J. 2020. An Attention-Based Graph Neural Network for Heterogeneous Structural Learning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 4132–4139. AAAI Press.

Huang, C.; Ren, X.; Tang, J.; Yin, D.; and Chawla, N. V. 2024. Large Language Models for Graphs: Progresses and Directions. In Chua, T.; Ngo, C.; Lee, R. K.; Kumar, R.; and Lauw, H. W., eds., *Companion Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, Singapore, May 13-17, 2024*, 1284–1287. ACM.

Huang, Q.; Ren, H.; Chen, P.; Krzmanc, G.; Zeng, D.; Liang, P.; and Leskovec, J. 2023. PRODIGY: Enabling In-context Learning Over Graphs. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Kong, L.; Feng, J.; Liu, H.; Huang, C.; Huang, J.; Chen, Y.; and Zhang, M. 2024. GOFA: A Generative One-For-All Model for Joint Graph Language Modeling. arXiv:2407.09709.

Li, C.; and Goldwasser, D. 2019. Encoding Social Information with Graph Convolutional Networks forPolitical Perspective Detection in News Media. In Korhonen, A.; Traum, D.; and Màrquez, L., eds., *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2594–2604. Florence, Italy: Association for Computational Linguistics.

Li, Q.; Han, Z.; and Wu, X. 2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In McIlraith, S. A.; and Weinberger, K. Q., eds., *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 3538–3545. AAAI Press.

Liu, H.; Feng, J.; Kong, L.; Liang, N.; Tao, D.; Chen, Y.; and Zhang, M. 2024. One For All: Towards Training One Graph Model For All Classification Tasks. In *The Twelfth International Conference on Learning Representations*.

Luo, Z.; Song, X.; Huang, H.; Lian, J.; Zhang, C.; Jiang, J.; Xie, X.; and Jin, H. 2024. GraphInstruct: Empowering Large Language Models with Graph Understanding and Reasoning Capability. *CoRR*, abs/2403.04483.

Lv, Q.; Ding, M.; Liu, Q.; Chen, Y.; Feng, W.; He, S.; Zhou, C.; Jiang, J.; Dong, Y.; and Tang, J. 2021. Are we really making much progress?: Revisiting, benchmarking and refining heterogeneous graph neural networks. In Zhu, F.; Ooi, B. C.; and Miao, C., eds., *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, 1150–1160. ACM.

Mernyei, P.; and Cangea, C. 2020. Wiki-CS: A Wikipedia-Based Benchmark for Graph Neural Networks. *CoRR*, abs/2007.02901.

Qiu, J.; Tang, J.; Ma, H.; Dong, Y.; Wang, K.; and Tang, J. 2018. DeepInf: Social Influence Prediction with Deep Learning. In Guo, Y.; and Farooq, F., eds., *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, 2110–2119. ACM.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Inui, K.; Jiang, J.; Ng, V.; and Wan, X., eds., *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, 3980–3990. Association for Computational Linguistics.

Sun, X.; Cheng, H.; Li, J.; Liu, B.; and Guan, J. 2023. All in One: Multi-Task Prompting for Graph Neural Networks. In Singh, A. K.; Sun, Y.; Akoglu, L.; Gunopulos, D.; Yan, X.; Kumar, R.; Ozcan, F.; and Ye, J., eds., *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, 2120–2131. ACM.

Tan, Y.; Lv, H.; Huang, X.; Zhang, J.; Wang, S.; and Yang, C. 2024. MuseGraph: Graph-oriented Instruction Tuning of Large Language Models for Generic Graph Mining. *CoRR*, abs/2403.04780.

Tang, J.; Yang, Y.; Wei, W.; Shi, L.; Su, L.; Cheng, S.; Yin, D.; and Huang, C. 2024. GraphGPT: Graph Instruction Tuning for Large Language Models. In Yang, G. H.; Wang, H.; Han, S.; Hauff, C.; Zuccon, G.; and Zhang, Y., eds., *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, 491–500. ACM.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Wang, H.; Feng, S.; He, T.; Tan, Z.; Han, X.; and Tsvetkov, Y. 2023a. Can Language Models Solve Graph Problems in Natural Language? In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Wang, X.; Bo, D.; Shi, C.; Fan, S.; Ye, Y.; and Yu, P. S. 2023b. A Survey on Heterogeneous Graph Embedding: Methods, Techniques, Applications and Sources. *IEEE Trans. Big Data*, 9(2): 415–436.

Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; and Yu, P. S. 2019. Heterogeneous Graph Attention Network. In Liu, L.; White, R. W.; Mantrach, A.; Silvestri, F.; McAuley, J. J.; Baeza-Yates, R.; and Zia, L., eds., *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, 2022–2032. ACM.

Yang, B.; Yih, W.; He, X.; Gao, J.; and Deng, L. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Yang, C.; Xiao, Y.; Zhang, Y.; Sun, Y.; and Han, J. 2022. Heterogeneous Network Representation Learning: A Unified Framework With Survey and Benchmark. *IEEE Trans. Knowl. Data Eng.*, 34(10): 4854–4873.

Yang, X.; Yan, M.; Pan, S.; Ye, X.; and Fan, D. 2023. Simple and efficient heterogeneous graph neural network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 10816–10824.

Yun, S.; Jeong, M.; Kim, R.; Kang, J.; and Kim, H. J. 2019. Graph Transformer Networks. In Wallach, H. M.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. B.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 11960–11970.

Zhang, C.; Song, D.; Huang, C.; Swami, A.; and Chawla, N. V. 2019. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 793–803.

Zhou, Z.; Shi, J.; Yang, R.; Zou, Y.; and Li, Q. 2023. SlotGAT: Slot-based Message Passing for Heterogeneous Graphs. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, 42644–42657. PMLR.

Zhu, S.; Zhou, C.; Pan, S.; Zhu, X.; and Wang, B. 2019. Relation structure-aware heterogeneous graph neural network. In *2019 IEEE international conference on data mining (ICDM)*, 1534–1539. IEEE.