Uni-GraphLLM: Unified Graph Large Language Model for Multi-Domain Multi-Task Graph Alignment Instruction Tuning

Haibo Chen
Tsinghua University
chb24@mails.tsinghua.edu.cn

Xin Wang*
Tsinghua University
xin_wang@tsinghua.edu.cn

Jiaheng Chao
Tsinghua University
chaojh26@mails.tsinghua.edu.cn

Ling Feng
Tsinghua University
fengling@tsinghua.edu.cn

Wenwu Zhu*
Tsinghua University
wwzhu@tsinghua.edu.cn

Abstract

Leveraging Graph Neural Networks (GNNs) as graph encoders and aligning the resulting representations with Large Language Models (LLMs) through alignment instruction tuning has become a mainstream paradigm for constructing Graph Language Models (GLMs), combining the generalization ability of LLMs with the structural modeling capacity of GNNs. However, existing GLMs that adopt GNNs as graph encoders largely overlook the problem of aligning GNN-encoded representations across domains and tasks with the LLM token space to obtain unified graph tokens, thereby limiting their ability to generalize across diverse graph data. To bridge this gap, we aim to incorporate a multi-domain, multi-task GNN encoder into GLMs and align its representations with LLMs to enable multi-domain, multi-task graph alignment instruction tuning. This alignment problem remains underexplored and poses two key challenges: 1) learning GNN-encoded representations that are simultaneously generalizable across domains and tasks and well aligned with textual semantics is difficult, due to substantial variations in graph structures, feature distributions, and supervision signals, together with the lack of textual-semantic alignment guidance in task-specific GNN training; 2) diverse graph data and task-specific instructions can exhibit different degrees of compatibility with the LLM token space during instruction tuning, leading to varying alignment difficulty and rendering a fixed alignment strategy suboptimal. To tackle these challenges, we propose **Uni-GraphLLM**, a **Uni**fied **Graph** **L**arge **L**anguage **M**odel that incorporates a multi-domain, multi-task GNN encoder to learn generalizable graph representations aligned with textual semantics, and then adaptively aligns these representations with the LLM. Specifically, we first develop a graph-text pair pretraining strategy with a tailored GNN encoder, trained on large-scale graph-text data spanning multiple domains and tasks to obtain generalizable representations naturally aligned with textual semantics. We further design a curriculum alignment tuning strategy that adaptively adjusts the alignment process by accounting for varying alignment difficulty across diverse graph data. Extensive

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

experiments demonstrate that Uni-GraphLLM consistently outperforms state-of-the-art baselines across graph datasets from different domains and tasks. Our code is available at https://github.com/haibo12/Uni-GraphLLM.

1 Introduction

Recent advances in Large Language Models (LLMs) have motivated the development of Graph Language Models (GLMs) [1, 2, 3] [4], which aim to extend the generalization and reasoning capabilities of LLMs to graph-structured data. Inspired by the success of Vision Language Models (VLMs) [5, 6, 7], existing GLMs typically follow a VLM-style two-stage architecture: a graph encoder first maps graph-structured data into graph representations, which are then aligned with the LLM token embedding space through alignment instruction tuning to produce graph tokens for downstream tasks. In this paradigm, Graph Neural Networks (GNNs) have become the dominant choice of graph encoders due to their strong ability to capture both structural and semantic information in graphs [8, 9, 10]. As a result, leveraging GNNs as graph encoders and aligning their representations with LLMs has become a mainstream paradigm for constructing GLMs, combining the expressive power of GNNs on graph data with the strong generalization capabilities of LLMs.

However, existing GLMs that use GNNs as graph encoders are typically designed around task- or domain-specific graph representation learning, with limited consideration of how GNN-encoded representations from diverse domains and tasks can be consistently aligned with the LLM token space to obtain unified graph tokens. This restricts their ability to generalize across diverse graph data. Such limited generalization stems from the inherent diversity of graph data: graph structures, feature distributions, and supervision signals often vary substantially across domains and tasks, making it difficult for GNNs to learn generalizable representations in a unified manner [11, 12] [13]. Moreover, the modality gap between GNN-encoded representations and textual semantics further complicates their alignment with LLMs [9, 10]. In addition, such variations in graph structures, feature distributions, and task-specific instructions also lead to varying levels of alignment difficulty across graph data, making it challenging for a unified alignment strategy to adapt effectively [14].

Motivated by these observations, this paper studies how to incorporate a multi-domain, multi-task GNN encoder into GLMs and align its representations with the LLM token space to enable multi-domain, multi-task graph alignment instruction tuning, producing unified graph tokens for diverse graph data. Despite its importance, this alignment problem remains underexplored and raises two key challenges: 1) **Generalizable text-aligned representation learning.** Learning GNN-encoded representations that are simultaneously generalizable across domains and tasks and well aligned with textual semantics is difficult, due to substantial variations in graph structures, feature distributions, and supervision signals, together with the lack of textual-semantic alignment guidance in task-specific GNN training. 2) **Varying alignment difficulty.** Diverse graph data and task-specific instructions can exhibit different degrees of compatibility with the LLM token space during instruction tuning, resulting in varying alignment difficulties and making a fixed alignment strategy suboptimal.

To tackle these challenges, we propose **Uni-GraphLLM**, a **Uni**fied **Graph** **L**arge **L**anguage **M**odel that incorporates a multi-domain, multi-task GNN encoder to learn generalizable graph representations aligned with textual semantics across domains and tasks, and adaptively aligns these representations with the LLM token space during instruction tuning. Specifically, we first propose a graph-text pair pretraining strategy, where a tailored GNN encoder is trained on large-scale graph-text datasets spanning multiple domains and tasks, enabling it to learn generalizable representations naturally aligned with textual semantics and thereby facilitating subsequent alignment with LLMs. Furthermore, we design a curriculum alignment tuning strategy that adaptively adjusts the alignment process by accounting for varying alignment difficulties induced by the diversity of graph data across domains and tasks, enabling more effective alignment of GNN representations with LLMs. Extensive experiments demonstrate that Uni-GraphLLM consistently outperforms state-of-the-art baselines across diverse graph domains and tasks. The contributions of this paper are summarized as follows:

- To the best of our knowledge, Uni-GraphLLM is the first work to jointly incorporate a shared multi-domain, multi-task GNN encoder into GLMs and explicitly align its representations with LLMs to produce unified graph tokens for diverse graph data, paving the way for GNN-encoder-based graph language models that generalize across domains and tasks.

- We propose a graph-text pair pretraining strategy for learning generalizable representations aligned with textual semantics, and a curriculum alignment tuning strategy for adapting the alignment process to varying difficulty across diverse graph data.
- We conduct extensive experiments across diverse graph domains and tasks, demonstrating that Uni-GraphLLM consistently outperforms state-of-the-art GLM baselines under both multi-domain multi-task learning and cross-domain/cross-task generalization settings.

2 Related Works

LLM for Graph via Graph-to-Text. With the strong capabilities of Large Language Models (LLMs) in natural language understanding and reasoning, a natural approach is to convert graphs into textual descriptions, *i.e.*, graph-to-text, and feed them into LLMs to perform graph-related tasks [15, 16, 17, 18, 19]. For instance, NLGraph [20] transforms graph structures into natural language problem descriptions and evaluates whether LLMs can directly solve classical graph reasoning tasks, such as connectivity, shortest path, and maximum flow, through textual inputs. Similarly, LLM4DyG [21] introduces a benchmark that encodes dynamic graph structures into natural language and designs a diverse set of tasks, including temporal link queries, path reasoning, and triadic closure, to systematically assess LLMs' ability to understand and reason over both structural and spatio-temporal information from text. However, this paradigm suffers from inherent limitations: due to the large scale and complex topology of graph data, it is difficult to faithfully represent an entire graph using plain text; moreover, encoding full graph information as textual input incurs substantial token overhead, leading to increased computational cost and limited scalability.

LLM for Graph via Graph-to-Token. In contrast to graph-to-text approaches, another line of work adopts a graph-to-token paradigm, which we refer to as Graph Language Models (GLMs) [22, 23, 24, 25, 26] in this paper. In this paradigm, a graph encoder transforms graph structures into compact continuous representations, which are aligned with the LLM token embedding space for downstream reasoning [27, 28]. Compared with verbose natural language descriptions, such graph tokens enable more efficient and scalable processing of graph data. For example, GraphGPT [8] aligns graph structural knowledge with LLMs through text-graph grounding and dual-stage instruction tuning, enabling generative graph reasoning. Similarly, LLaGA [29] adapts graph data into a structure-aware sequential format by reorganizing nodes and projecting graph representations into the LLM token embedding space, enabling unified sequence modeling over graph inputs. Owing to their strong capability in modeling graph-structured data, Graph Neural Networks (GNNs) have become the dominant choice of graph encoders, transforming graphs into expressive continuous representations [8, 9, 10, 30, 31, 32, 33]. However, existing GNN-based GLMs are typically trained on a single domain or task, and still face significant challenges in aligning multi-domain, multi-task GNN representations with the LLM token space to obtain unified graph tokens for diverse graph data.

3 Problem Formulation

In this section, we first define multi-domain, multi-task graph data and then describe the graph alignment instruction tuning process. Finally, we formulate the problem of multi-domain, multi-task graph alignment instruction tuning.

3.1 Multi-domain, Multi-task Graph Data

We consider a collection of graph datasets $\mathcal{D}$, where each dataset $D \in \mathcal{D}$ is treated as a distinct domain and is associated with a task type $t_D \in \mathcal{T}$, with $\mathcal{T}$ denoting the task-type space. The task type determines the required granularity of the graph representation (node-, edge-, or graph-level). Formally, dataset D is represented as a set of graph-task instances $\{(G_i, y_i)\}_{i=1}^{N_D}$, where G_i is a graph instance drawn from D, $y_i \in \mathcal{Y}_{t_D}$ is its task-specific label, and N_D is the number of instances in D. Each graph instance is defined as $G_i = (\mathcal{V}_i, \mathcal{E}_i, \mathbf{X}_i, \mathbf{E}_i)$, where $\mathcal{V}_i$ and $\mathcal{E}_i$ denote its node and edge sets, and $\mathbf{X}_i$, $\mathbf{E}_i$ denote the corresponding node and edge features.

3.2 Graph Alignment Instruction Tuning

For a single dataset D with task type t_D, graph alignment instruction tuning converts each graph-task instance (G_i, y_i) into a graph-conditioned instruction. Let $\hat{\mathbf{X}}_i$ denote the task-granularity graph representation obtained by applying the graph encoder parameterized by ϕ to G_i. A projector layer then maps $\hat{\mathbf{X}}_i$ into a sequence of graph tokens $\mathbf{z}_i = \mathrm{Project}_\theta(\hat{\mathbf{X}}_i)$. The graph-conditioned instruction is then constructed as $I_i = [\mathbf{q}_i; \mathbf{z}_i]$, where $\mathbf{q}_i$ denotes the natural language instruction describing the task and $\mathbf{z}_i$ denotes the graph token sequence. The target output $\mathbf{t}_i$ is derived from the task label y_i. The instruction tuning objective follows the standard next-token prediction paradigm, where the LLM generates $\mathbf{t}_i$ autoregressively conditioned on the instruction $\mathbf{q}_i$ and graph tokens $\mathbf{z}_i$:

$$\mathcal{L}_D = -\sum_{i=1}^{N_D} \log P(\mathbf{t}_i \mid \mathbf{q}_i, \mathbf{z}_i; \phi, \theta, \psi), \tag{1}$$

where ϕ denotes the parameters of the graph encoder, θ denotes the parameters of the projector layer, and ψ denotes the parameters of the LLM.

3.3 Multi-domain, Multi-task Graph Alignment Instruction Tuning

This work studies how to incorporate a multi-domain, multi-task GNN encoder into GLMs and jointly align its representations from multiple datasets with LLMs to produce unified graph tokens for diverse graph data. Since different graph tasks require different representation granularities, a unified encoder should be able to produce node-level, edge-level, and graph-level representations, denoted as $\mathbf{X}_i^{\text{node}}(v)$, $\mathbf{X}_i^{\text{edge}}(u, v)$, and $\mathbf{X}_i^{\text{graph}}$, respectively. For each dataset $D \in \mathcal{D}$ with task type t_D, let $* \in \{\text{node}, \text{edge}, \text{graph}\}$ denote the required representation granularity. A multi-scale GNN encoder $\mathrm{GNN}_\phi^{\text{multi}}(\cdot)$ extracts the task-specific representation $\mathbf{X}_i^* = \mathrm{GNN}_\phi^{\text{multi}}(G_i, *)$, which is then projected into the LLM token space to obtain graph tokens $\mathbf{z}_i^* = \mathrm{Project}_\theta(\mathbf{X}_i^*)$.

The goal of multi-domain, multi-task graph representation alignment is to learn a shared GNN encoder and projector that can produce graph tokens consistently aligned with LLMs across all datasets in $\mathcal{D}$. Accordingly, the overall alignment objective aggregates the instruction tuning losses over multiple datasets:

$$\mathcal{L}_{\text{align}} = \sum_{D \in \mathcal{D}} \mathcal{L}_D = -\sum_{D \in \mathcal{D}} \sum_{i=1}^{N_D} \log P(\mathbf{t}_i \mid \mathbf{q}_i, \mathbf{z}_i^*; \phi, \theta, \psi), \tag{2}$$

where $\mathcal{L}_D$ is the alignment loss on dataset D. This requires the GNN encoder to learn representations that are both generalizable across domains and tasks and amenable to alignment with LLMs, while the projector should adapt the alignment process to graph data with varying alignment difficulties.

4 Method

In this section, we present Uni-GraphLLM in detail. It consists of two key components: graph-text pair pretraining and curriculum alignment tuning. The overall framework is illustrated in Figure 1.

4.1 Graph-Text Pair Pretraining

To learn generalizable, text-aligned representations and facilitate subsequent alignment with LLMs, we propose a graph-text pair pretraining strategy. Specifically, we construct large-scale graph-text pairs across diverse domains and tasks, design a multi-scale GNN encoder to capture representations at different granularities, and pretrain it using a contrastive objective with reweighting to capture multi-domain semantic differences and enhance generalization.

Graph-Text Pair Construction. Initially, we construct a large-scale graph-text pair dataset by collecting graph data from multiple domains and tasks, and pairing each graph instance with a corresponding textual description. Formally, the graph-text pair dataset is defined as $\mathcal{D}_{\text{gt}} = \{(G_i, T_i)\}_{i=1}^N$, where G_i is a graph instance and T_i is its textual description. Notably, this construction does not require task labels y_i, making it well-suited for real-world pretraining scenarios where large-scale graph data often lacks annotations.

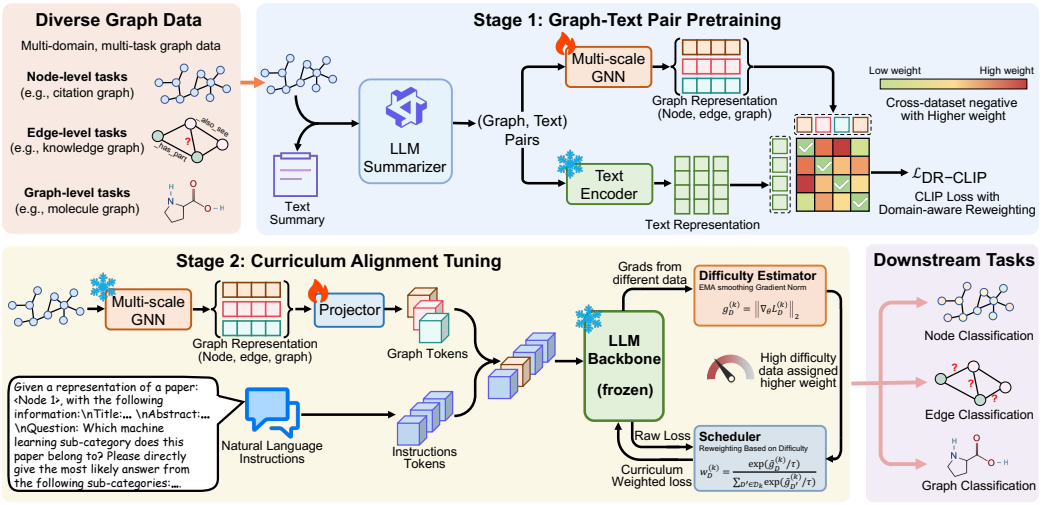


Figure 1: Overall framework of Uni-GraphLLM. *Stage 1: Graph-Text Pair Pretraining.* We construct large-scale graph-text pairs across multiple domains and tasks, encode each graph using a multi-scale GNN encoder to produce its task-required node-, edge-, or graph-level representation in a shared space, and train the encoder with a domain-aware reweighted contrastive objective that explicitly accounts for both inter-domain and intra-domain semantic differences. *Stage 2: Curriculum Alignment Tuning.* During instruction tuning, we estimate domain-level alignment difficulty online from per-domain gradient statistics and adaptively reweight the training objective to focus more on harder domains, leading to more balanced and effective alignment between GNN representations and the LLM.

Inspired by GraphCLIP [34], we use an LLM (Qwen3-8B [35]) as a preprocessing tool to generate textual descriptions for graph instances. Given a graph instance $G_i = (\mathcal{V}_i, \mathcal{E}_i, \mathbf{X}_i, \mathbf{E}_i)$, the LLM summarizes its raw textual features $(\mathbf{X}_i, \mathbf{E}_i)$ while incorporating structural information $(\mathcal{V}_i, \mathcal{E}_i)$, including node and edge attributes, to produce a coherent natural language description. We design dataset-specific prompts to generate descriptions tailored to the characteristics of different datasets; the full set of summarization prompts is provided in Appendix C.1. The resulting descriptions capture both structural and semantic information of each graph instance, providing rich supervision for subsequent graph-text pair pretraining.

Multi-scale Graph Representation. Since different tasks require graph representations at different scales, we design a multi-scale GNN encoder that produces representations at varying levels of granularity within a shared embedding space. First, we employ a GNN to extract node-level representations, and then derive a graph-level representation via a pooling operation. Given a graph $G_i = (\mathcal{V}_i, \mathcal{E}_i, \mathbf{X}_i, \mathbf{E}_i)$, the representations are computed as:

$$\mathbf{H}_i^{\text{node}} = \text{GNN}(G_i), \quad \mathbf{H}_i^{\text{graph}} = \text{Pooling}(\mathbf{H}_i^{\text{node}}), \tag{3}$$

where $\mathbf{H}_i^{\text{node}} \in \mathbb{R}^{|\mathcal{V}_i| \times d}$ and $\mathbf{H}_i^{\text{graph}} \in \mathbb{R}^d$ denote the node-level and graph-level representations, respectively, and d is the representation dimension.

To enable a single GNN to produce representations at different levels of granularity within the same space, we further define task-specific aggregation functions that combine node- and graph-level representations into appropriate task-scale representations.

- *Node-level Tasks.* For node-level tasks, the representation of a target node v combines its local node-level embedding with the global graph context:

$$\mathbf{X}_i^{\text{node}}(v) = \text{MLP}\left([\mathbf{H}_i^{\text{node}}(v) \,\|\, \mathbf{H}_i^{\text{graph}}]\right). \tag{4}$$

- *Edge-level Tasks.* For edge-level tasks, we derive an edge representation by aggregating its endpoint node embeddings (u, v) and incorporating global graph context:

$$\mathbf{X}_i^{\text{edge}}(u, v) = \text{MLP}\left(\left[\frac{\mathbf{H}_i^{\text{node}}(u) + \mathbf{H}_i^{\text{node}}(v)}{2} \,\|\, \mathbf{H}_i^{\text{graph}}\right]\right), \quad (u, v) \in \mathcal{E}_i. \tag{5}$$

- *Graph-level Tasks.* For graph-level tasks, we apply the same aggregation module to the graph-level representation to ensure consistency in both dimensionality and embedding space:

$$\mathbf{X}_i^{\text{graph}} = \text{MLP}\left([\mathbf{H}_i^{\text{graph}} \,\|\, \mathbf{H}_i^{\text{graph}}]\right). \tag{6}$$

We denote the unified multi-scale GNN encoder as $\text{GNN}_\phi^{\text{multi}}(\cdot)$, which produces representations $\mathbf{X}_i^*$ at different levels of granularity with $* \in \{\text{node}, \text{edge}, \text{graph}\}$. This unified design ensures that representations at different granularities share a consistent dimensionality and embedding space, facilitating their alignment with LLMs across diverse tasks.

Domain-aware Reweighting. A natural way to train the GNN encoder on graph-text pairs is to adopt a contrastive learning objective similar to CLIP [36], which pulls matched graph-text pairs closer and pushes mismatched ones apart in a shared embedding space. However, since graph-text pairs are drawn from multiple domains, each corresponding to a distinct dataset, treating all negative samples equally is suboptimal: negatives from the same domain as the anchor are typically semantically closer than those from other domains, and uniform weighting may therefore over-penalize intra-domain negatives while under-utilizing inter-domain negatives. To address this issue, we introduce a domain-aware reweighting strategy that assigns larger weights to negatives from more distant domains and smaller weights to intra-domain or nearby-domain negatives, thereby encouraging stronger cross-domain separation while preserving fine-grained intra-domain semantics.

Initially, we employ GraphSAGE [37] as the multi-scale GNN encoder to extract graph representations $\mathbf{X}_i^*$ (Eqs. 4–6), and use Sentence-BERT [38] as the text encoder to obtain text representations $\mathbf{T}_i$ from the textual descriptions T_i:

$$\mathbf{T}_i = \text{Pooling}\big(\text{Sentence-BERT}(T_i)\big). \tag{7}$$

Given graph-text representation pairs $\{(\mathbf{X}_i^*, \mathbf{T}_i)\}_{i=1}^N$ derived from paired inputs $\{(G_i, T_i, d_i)\}_{i=1}^N$, where $d_i \in \mathcal{D}$ denotes the domain (dataset) identifier of the i-th pair, we compute their cosine similarities and define the bidirectional similarity matrices as:

$$\mathbf{S}_{ij}^{g\to t} = \frac{\mathbf{X}_i^{*\top}\mathbf{T}_j}{\|\mathbf{X}_i^*\|_2\|\mathbf{T}_j\|_2}, \quad \mathbf{S}_{ij}^{t\to g} = \frac{\mathbf{T}_i^{\top}\mathbf{X}_j^*}{\|\mathbf{T}_i\|_2\|\mathbf{X}_j^*\|_2}. \tag{8}$$

To construct domain-aware weights, we compute domain-level centers using the initial graph features $\mathbf{X}_i$ of $G_i = (\mathcal{V}_i, \mathcal{E}_i, \mathbf{X}_i, \mathbf{E}_i)$ and the text representations $\mathbf{T}_i$ of the corresponding textual descriptions. These centers serve as proxies for domain similarity. For each domain $a \in \mathcal{D}$, we sample up to 1000 instances to compute graph and text centers, denoted as $\mathbf{c}_a^g$ and $\mathbf{c}_a^t$, respectively. Then, for any pair of domains $a, b \in \mathcal{D}$, we define the normalized inter-domain distances:

$$[M_g]_{ab} = \frac{1-\cos(\mathbf{c}_a^g, \mathbf{c}_b^g)}{\max_{u,v}(1-\cos(\mathbf{c}_u^g, \mathbf{c}_v^g))}, \quad [M_t]_{ab} = \frac{1-\cos(\mathbf{c}_a^t, \mathbf{c}_b^t)}{\max_{u,v}(1-\cos(\mathbf{c}_u^t, \mathbf{c}_v^t))}. \tag{9}$$

We then construct domain-aware weights as:

$$W_{ab}^g = 1 + [M_g]_{ab}, \quad W_{ab}^t = 1 + [M_t]_{ab}. \tag{10}$$

Building upon the standard contrastive formulation, we incorporate these weights into the loss. For each anchor i, (i, i) forms the positive pair, while $j \neq i$ correspond to negative samples. The weighted contrastive losses are defined as:

$$\mathcal{L}_{g\to t} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp(\mathbf{S}_{ii}^{g\to t})}{\exp(\mathbf{S}_{ii}^{g\to t}) + \sum_{j\neq i} W_{d_i,d_j}^g \exp(\mathbf{S}_{ij}^{g\to t})}, \tag{11}$$

$$\mathcal{L}_{t\to g} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp(\mathbf{S}_{ii}^{t\to g})}{\exp(\mathbf{S}_{ii}^{t\to g}) + \sum_{j\neq i} W_{d_i,d_j}^t \exp(\mathbf{S}_{ij}^{t\to g})}, \tag{12}$$

$$\mathcal{L}_{\text{DR-CLIP}} = \frac{1}{2}\left(\mathcal{L}_{g\to t} + \mathcal{L}_{t\to g}\right). \tag{13}$$

This design assigns larger weights to negatives drawn from more distant domains, encouraging stronger inter-domain separation while preserving fine-grained intra-domain alignment. The overall pretraining objective is $\mathcal{L}_{\text{DR-CLIP}}$, which provides generalizable, text-aligned graph representations for the subsequent instruction tuning stage.

4.2 Curriculum Alignment Tuning

To adaptively adjust the alignment process to varying difficulties induced by the diversity of graph data across domains and tasks, we propose a curriculum alignment tuning strategy. It estimates domain-level alignment difficulty online during instruction tuning and accordingly reweights the training objective, encouraging the model to focus more on challenging domains while maintaining exposure to easier ones for more balanced and effective alignment.

Alignment Difficulty Estimation. Due to memory and compute constraints, it is infeasible to re-estimate the alignment difficulty of every domain over the entire dataset at each optimization step. Instead, we adopt an online estimation strategy with Exponential Moving Average (EMA) [39] smoothing that dynamically combines statistics from the current mini-batch with historical estimates.

At optimization step k, let the mini-batch be $\mathcal{B}_k$. We group samples by their originating datasets, each corresponding to a domain, yielding the set of active domains $\mathcal{D}_k$. We compute the per-instance loss L_i according to Eq. (1), and aggregate it to obtain the dataset-level loss for each dataset $D \in \mathcal{D}_k$:

$$L_D^{(k)} = \frac{1}{|\mathcal{B}_k^D|} \sum_{i \in \mathcal{B}_k^D} L_i, \tag{14}$$

where $\mathcal{B}_k^D \subset \mathcal{B}_k$ denotes the subset of samples from dataset D. Then, we quantify the alignment difficulty of each domain (i.e., dataset D) using the gradient norm:

$$g_D^{(k)} = \left\| \nabla_\theta L_D^{(k)} \right\|_2, \tag{15}$$

where θ denotes the projector parameters. During alignment tuning, both the pretrained GNN encoder and the LLM are frozen, and only the projector is optimized. Because the projector's sole role is to map graph representations into the LLM token space, its gradient directly measures the domain-specific update demanded from this mapping. Motivated by prior work that uses gradient-based quantities to characterize sample importance or difficulty [40, 41, 42], we use a larger projector gradient norm as an operational proxy for stronger current alignment difficulty. To obtain stable estimates, we employ a two-stage smoothing scheme. Because early-stage gradients are often highly noisy, directly applying EMA may propagate this noise. We therefore use a running mean as a warmup estimator before switching to EMA. Let T denote the total number of training steps and $\rho \in [0, 1]$ the warmup ratio, with $T_w = \lfloor \rho T \rfloor$. The smoothed difficulty score is defined as:

$$\tilde{g}_D^{(k)} = \begin{cases} \mu_D^{(k)} = \frac{\sum_{u=1}^{k} g_D^{(u)} \mathbb{I}[D \in \mathcal{D}_u]}{\sum_{u=1}^{k} \mathbb{I}[D \in \mathcal{D}_u]}, & k < T_w, \\ \beta\, \tilde{g}_D^{(k-1)} + (1-\beta)\, g_D^{(k)}, & k \geq T_w, \end{cases} \tag{16}$$

where β is the EMA momentum. For a newly observed dataset, we initialize $\tilde{g}_D$ with $\mu_D^{(k)}$ at its first occurrence. Importantly, the running-mean and EMA updates are performed only for active domains $D \in \mathcal{D}_k$ that appear in the current mini-batch. For inactive domains $D \notin \mathcal{D}_k$, we keep their previous estimates unchanged, i.e., $\tilde{g}_D^{(k)} = \tilde{g}_D^{(k-1)}$, rather than treating their missing gradients as zero. This procedure yields robust estimates of domain-level alignment difficulty.

Alignment Curriculum Schedule. Based on the estimated difficulty, we adaptively reweight the active domains in the current mini-batch via a temperature-scaled softmax:

$$w_D^{(k)} = \frac{\exp\left(\tilde{g}_D^{(k)}/\tau\right)}{\sum_{D' \in \mathcal{D}_k} \exp\left(\tilde{g}_{D'}^{(k)}/\tau\right)}, \tag{17}$$

where τ is a temperature hyperparameter. This formulation prioritizes domains with higher estimated difficulty, allocating more learning capacity to challenging data while still maintaining exposure to easier ones [43, 44]. Compared to uniform training, such adaptive reweighting mitigates the dominance of easy domains and reduces the risk of underfitting harder ones. Finally, the training objective at step k:

$$\mathcal{L}_k = \sum_{D \in \mathcal{D}_k} w_D^{(k)}\, L_D^{(k)}. \tag{18}$$

This curriculum-driven objective dynamically adjusts the optimization weights assigned to different domains over the course of training, resulting in more balanced and effective alignment across diverse domains and tasks. During this stage, we freeze both the pretrained GNN encoder and the LLM, and optimize only the projector parameters. This preserves the generalizable graph representations learned during graph-text pair pretraining, allowing the curriculum strategy to focus on adapting the alignment module to the LLM. The full algorithm is provided in Appendix B.

5 Experiments

5.1 Experimental Setup

Datasets. We conduct experiments on diverse datasets spanning multiple domains and task types. For node-level tasks, we consider citation networks, including *Cora*, *PubMed*, and *Arxiv*, as well as the web hyperlink network *Wiki-CS*. For edge-level tasks, we adopt two knowledge graph datasets, *WN18RR* and *FB15K237(10-way)*. Since *FB15K237* contains 237 relation labels, including all candidates in each instruction would make the input overly long; we therefore use 10-way candidate sampling, consisting of the correct label and 9 randomly sampled negative labels. For graph-level tasks, we use two molecular datasets, *ChemHIV* and *ChemPCBA*. Details of dataset statistics and preprocessing are provided in Appendix D.1.

Baselines. We compare Uni-GraphLLM with pure LLM baselines and several state-of-the-art graph language models, including GraphGPT [8], LLaGA [29], GOFA [45], and TEA-GLM [10]. To ensure a fair comparison, we use LLM backbones of comparable scale across all methods. Detailed descriptions of the baselines and their configurations are provided in Appendix D.2.

5.2 Main Results

To evaluate whether large-scale pretraining and alignment enable our model to generalize effectively across domains and tasks, we consider three evaluation settings: multi-domain multi-task learning, cross-domain generalization, and cross-task generalization.

Table 1: Performance comparison of different methods on node classification, edge classification, and graph classification tasks under multi-domain and multi-task learning. The highest result is **bold**, and the second highest result is underlined.

	Node Classification(ACC%)				**Edge Classification(ACC%)**		**Graph Classification(AUC%)**		
Method	Cora	PubMed	Wiki-CS	Arxiv	WN18RR	FB15K237	ChemPCBA	ChemHIV	***Avg.***
Vicuna-7B	56.67	78.92	62.34	43.75	34.84	56.01	46.47	30.36	51.17
LLaMA-2-7B	29.45	25.36	22.22	6.33	29.16	20.60	51.12	50.09	29.29
GraphGPT	57.59	73.45	49.20	67.63	N.S.	N.S.	N.S.	N.S.	61.97
LLaGA	<u>74.71</u>	68.54	<u>69.16</u>	<u>73.83</u>	N.S.	N.S.	N.S.	N.S.	71.56
GOFA	63.88	64.72	66.77	66.11	21.95	54.41	60.61	49.78	56.03
TEA-GLM	62.38	<u>85.13</u>	55.19	66.81	<u>92.85</u>	<u>91.85</u>	<u>80.40</u>	<u>67.13</u>	<u>75.22</u>
Ours	**76.98**	**90.49**	**76.65**	**74.22**	**93.52**	**91.96**	**81.69**	**70.09**	**81.95**

Multi-domain and multi-task learning. We first assess the effectiveness of jointly learning from multiple domains and task types. Specifically, we pretrain the graph encoder on the full collection of datasets spanning diverse domains and tasks in a label-free manner, perform instruction tuning of the projector layer using the combined training sets from all datasets, and directly evaluate the resulting model on each test split without dataset-specific fine-tuning.

As shown in Table 1, pure LLM baselines perform substantially worse than GLM methods on average (e.g., Vicuna-7B: 51.17% and LLaMA-2-7B: 29.29%), highlighting the importance of explicitly incorporating graph structural information via a dedicated graph encoder. Among GLMs, Uni-GraphLLM achieves the best average performance (81.95%), outperforming the strongest baseline, TEA-GLM (75.22%), by 6.73 percentage points. It delivers consistent gains across node-level, edge-level, and graph-level classification tasks and attains the best results on all datasets. These results validate the effectiveness of the proposed method in jointly learning from multiple domains and task types and generalizing effectively across domains and tasks.

Cross-domain and cross-task generalization. For cross-domain generalization, we evaluate whether the pretrained GNN encoder can support generalization across domains when instruction tuning is performed on only a single dataset. Specifically, we use the GNN encoder obtained from multi-dataset graph-text pair pretraining, perform instruction tuning only on Arxiv, a node-level dataset, and evaluate the resulting model on other node-level datasets from different domains, including Cora, PubMed, and Wiki-CS. For cross-task generalization, we further assess whether the same pretrained GNN encoder can transfer across task types under the same single-dataset instruction

tuning setting. Specifically, we again perform instruction tuning only on Arxiv, and evaluate the model on edge-level datasets, including WN18RR and FB15K237.

As shown in Table 2, Uni-GraphLLM substantially improves cross-domain generalization, achieving 55.17%, 77.31%, and 63.93% on Cora, PubMed, and Wiki-CS, respectively, outperforming all baselines. It also demonstrates strong cross-task transfer, especially on FB15K237 (55.27%). These results suggest that large-scale graph-text pair pretraining equips the GNN encoder with generalizable representations, enabling effective cross-domain and cross-task generalization.

Table 2: Performance comparison under cross-domain and cross-task settings. The best results are in **bold**, and the second best are underlined.

	Cross-domain(ACC%)			Cross-task(ACC%)	
Method	Cora	PubMed	Wiki-CS	WN18RR	FB15K237
GraphGPT	31.53	38.99	37.78	-	-
LLaGA	13.01	25.74	5.64	-	-
GOFA	16.30	51.14	40.11	18.79	16.93
TEA-GLM	43.33	34.33	36.69	22.65	11.72
Ours	**55.17**	**77.31**	**63.93**	**24.12**	**55.27**

Compatibility with different LLM backbones. To verify that Uni-GraphLLM does not rely on a specific LLM, we further evaluate it with Qwen3-8B under the same multi-domain, multi-task setting. As shown in Table 3, Uni-GraphLLM remains consistently effective with Qwen3-8B and improves the overall average from 81.95% to 84.10%, including gains on six of the eight datasets. These results demonstrate that the proposed framework is compatible with different LLM backbones and that its effectiveness is not specific to Vicuna.

Table 3: Performance of Uni-GraphLLM with different LLM backbones under multi-domain, multi-task learning. We report accuracy for node- and edge-level tasks and AUC for graph-level tasks.

LLM Backbone	Cora	PubMed	Wiki-CS	Arxiv	WN18RR	FB15K237	ChemPCBA	ChemHIV	Avg.
Vicuna-7B-v1.5	76.98	90.49	**76.65**	**74.22**	93.52	91.96	81.69	70.09	81.95
Qwen3-8B	**79.98**	**92.01**	75.94	72.48	**94.32**	**95.87**	**86.70**	**75.51**	**84.10**

Alignment difficulty proxy. To further validate the choice of difficulty measure, we replace the projector gradient norm with raw domain loss while keeping the backbone, training data, and scheduler unchanged. As shown in Table 4, using the loss leads to lower average performance across datasets, supporting the gradient norm as a more effective proxy for alignment difficulty.

Table 4: Comparison of domain-level alignment difficulty proxies on five representative datasets. The average is computed over the five reported datasets.

Proxy	Cora	PubMed	Wiki-CS	Arxiv	WN18RR	Avg.
Gradient norm	**76.98**	90.49	**76.65**	**74.22**	**93.52**	**82.37**
Loss	75.34	**90.65**	73.64	69.80	93.49	80.58

5.3 Ablation Study

To verify the effectiveness of the proposed components, we conduct ablation studies to compare the full model with ablated versions: 1) **w/o pre**: we remove the graph-text pair pretraining, where the GNN encoder is trained along with the projector layer during instruction tuning; 2) **w/o rew**: we remove the domain-aware reweighting during graph-text pair pretraining, treating all negatives equally; 3) **w/o cur**: we remove the curriculum strategy during instruction tuning.

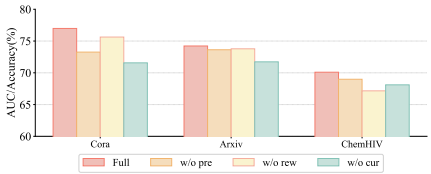


Figure 2: Performance comparison between the full model and different ablated versions.

The results are shown in Figure 2. We make the following observations. i) **w/o pre** exhibits a performance drop across datasets, indicating that large-scale graph-text pair pretraining provides a crucial initialization for learning generalizable and text-aligned graph representations. ii) **w/o rew** consistently underperforms the full model, suggesting that the proposed domain-aware reweighting helps the encoder better distinguish semantically distant inter-domain negatives while preserving intra-domain structure, thereby improving the quality of GNN representations for different domains. iii) **w/o cur** also leads to noticeable degradation, implying that difficulty-aware curriculum reweighting during alignment is important for handling diverse datasets and improving overall alignment quality.

5.4 Time and Resource Consumption

Graph-text pair pretraining is the most time-consuming stage, but it is performed only once, and the resulting encoder is reused for all subsequent alignment and evaluation. The alignment stage benefits clearly from distributed training: with the same workload, eight GPUs finish roughly four times faster than a single GPU. At inference time, the cost is mainly determined by the size of the test set and the input graphs, and the curriculum scheduler is not involved, so it adds no extra overhead. Compared with TEA-GLM, Uni-GraphLLM requires more inference memory, which is offset by its much stronger cross-domain generalization (Table 2). Stage-wise statistics are reported in Table 5, and comparisons with baselines are provided in Appendix D.4.

Table 5: Stage-wise computational cost of Uni-GraphLLM. Pretraining time is reported per epoch, alignment time is for the fixed one-epoch training schedule, and inference time covers each complete test split.

Stage / Dataset	**# GPUs**	**Batch Size**	**Time**	**Peak Memory / GPU**
Graph-text pair pretraining	6	4,096 (global)	5,861 s/epoch	20.59 GB
Alignment tuning (single-GPU)	1	3	23 h 33 min	65.5 GB
Alignment tuning (eight-GPU)	8	4/GPU	5 h 27 min	61.5 GB
Inference: Cora	1	40	225.5 s	47.7 GB
Inference: PubMed	1	40	2,856.5 s	50.1 GB
Inference: Wiki-CS	1	20	2,035.0 s	48.3 GB
Inference: Arxiv	1	20	7,078.9 s	35.1 GB
Inference: WN18RR	1	40	134.7 s	21.9 GB
Inference: FB15K237	1	40	2,205.5 s	24.7 GB
Inference: ChemPCBA	1	40	1,418.7 s	26.3 GB
Inference: ChemHIV	1	40	269.7 s	27.0 GB

6 Conclusion

In this paper, we study how to incorporate a multi-domain, multi-task GNN encoder into GLMs and align its representations with the LLM token space to produce unified graph tokens for diverse graph data. To this end, we propose Uni-GraphLLM, a unified graph large language model that leverages multi-domain, multi-task graph-text pair pretraining to learn generalizable graph representations aligned with the textual modality, together with a curriculum alignment tuning mechanism that adaptively aligns GNN representations with LLMs according to per-domain alignment difficulty. Extensive experiments under multi-domain multi-task learning as well as cross-domain and cross-task generalization settings demonstrate that Uni-GraphLLM consistently outperforms state-of-the-art GLM baselines. A limitation of our current approach is that it focuses on conventional graph task types (node, edge, and graph classification) over a moderate collection of datasets, without explicitly handling more complex graph reasoning scenarios. Scaling the pretraining corpus to larger and more diverse graph-text collections and systematically studying how the resulting gains transfer to downstream tasks are important directions for future work.

Acknowledgments and Disclosure of Funding

Funding in direct support of this work: Beijing National Research Center for Information Science and Technology under Grant No. BNR2026TD03005. Competing interests: The authors declare no competing interests.

References

[1] X. He, Y. Tian, Y. Sun, N. V. Chawla, T. Laurent, Y. LeCun, X. Bresson, and B. Hooi, "G-retriever: Retrieval-augmented generation for textual graph understanding and question answering," *Advances in Neural Information Processing Systems*, vol. 37, pp. 132 876–132 907, 2024.

[2] R. Ye, C. Zhang, R. Wang, S. Xu, and Y. Zhang, "Language is all a graph needs," in *Findings of the association for computational linguistics: EACL 2024*, 2024, pp. 1955–1973.

[3] Z. Chai, T. Zhang, L. Wu, K. Han, X. Hu, X. Huang, and Y. Yang, "GraphLLM: Boosting graph reasoning ability of large language model," *IEEE Transactions on Big Data*, vol. 12, no. 2, pp. 475–483, 2026.

[4] X. Wang, Z. Zhang, L. Xiao, H. Chen, C. Ge, and W. Zhu, "Towards multimodal graph large language model," *Science China Information Sciences*, vol. 68, no. 11, p. 213101, 2025.

[5] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," *Advances in neural information processing systems*, vol. 36, pp. 34 892–34 916, 2023.

[6] J. Li, D. Li, S. Savarese, and S. Hoi, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," in *International conference on machine learning*. PMLR, 2023, pp. 19 730–19 742.

[7] H. Liu, C. Li, Y. Li, and Y. J. Lee, "Improved baselines with visual instruction tuning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 26 296–26 306.

[8] J. Tang, Y. Yang, W. Wei, L. Shi, L. Su, S. Cheng, D. Yin, and C. Huang, "Graphgpt: Graph instruction tuning for large language models," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024, pp. 491–500.

[9] M. Zhang, M. Sun, P. Wang, S. Fan, Y. Mo, X. Xu, H. Liu, C. Yang, and C. Shi, "Graphtranslator: Aligning graph model to large language model for open-ended tasks," in *Proceedings of the ACM Web Conference 2024*, 2024, pp. 1003–1014.

[10] D. Wang, Y. Zuo, F. Li, and J. Wu, "Llms as zero-shot graph learners: Alignment of gnn representations with llm token embeddings," *Advances in neural information processing systems*, vol. 37, pp. 5950–5973, 2024.

[11] H. Liu, J. Feng, L. Kong, N. Liang, D. Tao, Y. Chen, and M. Zhang, "One for all: Towards training one graph model for all classification tasks," in *The Twelfth International Conference on Learning Representations*, 2024.

[12] Y. Sun, Y. Yang, X. Feng, Z. Wang, H. Zhong, C. Wang, and L. Chen, "Handling feature heterogeneity with learnable graph patches," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, 2025, pp. 1313–1324.

[13] H. Chen, X. Wang, Z. Zhang, H. Li, L. Feng, and W. Zhu, "AutoGFM: Automated graph foundation model with adaptive architecture customization," in *Proceedings of the 42nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 267, 2025, pp. 9295–9315.

[14] H. P. Wang, S. Liu, R. Wei, and P. Li, "Generalization principles for inference over text-attributed graphs with large language models," in *Forty-second International Conference on Machine Learning*, 2025. [Online]. Available: https://openreview.net/forum?id=dfOqiHuklY

[15] Y. Tan, Z. Zhou, H. Lv, W. Liu, and C. Yang, "Walklm: A uniform language model fine-tuning framework for attributed graph embedding," *Advances in neural information processing systems*, vol. 36, pp. 13 308–13 325, 2023.

[16] J. Tang, Q. Zhang, Y. Li, N. Chen, and J. Li, "GraphArena: Evaluating and exploring large language models on graph computation," in *The Thirteenth International Conference on Learning Representations*, 2025.

[17] X. Dai, H. Qu, Y. Shen, B. Zhang, Q. Wen, W. Fan, D. Li, J. Tang, and C. Shan, "How do large language models understand graph patterns? a benchmark for graph pattern comprehension," in *The Thirteenth International Conference on Learning Representations*, 2025.

[18] J. Wang, J. Wu, Y. Hou, Y. Liu, M. Gao, and J. McAuley, "Instructgraph: Boosting large language models via graph-centric instruction tuning and preference alignment," in *Findings of the Association for Computational Linguistics: ACL 2024*, 2024, pp. 13 492–13 510.

[19] S. Yu, Y. Wang, R. Li, G. Liu, Y. Shen, S. Ji, B. Li, F. Han, X. Zhang, and F. Xia, "Graph2text or graph2token: A perspective of large language models for graph learning," *ACM Transactions on Information Systems*, vol. 44, no. 3, pp. 1–49, 2026.

[20] H. Wang, S. Feng, T. He, Z. Tan, X. Han, and Y. Tsvetkov, "Can language models solve graph problems in natural language?" *Advances in Neural Information Processing Systems*, vol. 36, pp. 30 840–30 861, 2023.

[21] Z. Zhang, X. Wang, Z. Zhang, H. Li, Y. Qin, and W. Zhu, "Llm4dyg: Can large language models solve spatial-temporal problems on dynamic graphs?" in *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, 2024, pp. 4350–4361.

[22] B. Perozzi, B. Fatemi, D. Zelle, A. Tsitsulin, M. Kazemi, R. Al-Rfou, and J. Halcrow, "Let your graph do the talking: Encoding structured data for llms," *arXiv preprint arXiv:2402.05862*, 2024.

[23] Y. Chen, Q. Yao, J. Zhang, J. Cheng, and Y. Bian, "Hierarchical graph tokenization for molecule-language alignment," in *Forty-second International Conference on Machine Learning*, 2025.

[24] W. Guo, X. Wang, J. Chen, L. Guo, Z. Li, and Z. Chen, "ReaLM: Residual quantization bridging knowledge graph embeddings and large language models," in *Proceedings of the ACM Web Conference 2026*, 2026, pp. 3996–4007.

[25] Z. Zhang, X. Wang, M. Zhang, J. Tan, and C. Shi, "Toward graph-tokenizing large language models with reconstructive graph instruction tuning," in *Proceedings of the ACM Web Conference 2026*, 2026, pp. 430–441.

[26] Z. Wan, B. Wang, K. Fang, and B. Wu, "Tgca-llm: Time-aware graph-text contrastive alignment for enhancing llms in temporal knowledge graph completion," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 40, no. 18, pp. 15 806–15 814, Mar. 2026.

[27] R. Wang, M. Yang, and Y. Shen, "Graph2token: Make llms understand molecule graphs," in *ICML 2024 Workshop on Accessible and Efficient Foundation Models for Biological Discovery*, 2024.

[28] D. Wang, Y. Zuo, G. Lu, and J. Wu, "Unigte: Unified graph–text encoding for zero-shot generalization across graph tasks and domains," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[29] R. Chen, T. Zhao, A. K. Jaiswal, N. Shah, and Z. Wang, "Llaga: Large language and graph assistant," in *International Conference on Machine Learning*. PMLR, 2024, pp. 7809–7823.

[30] X. Wang, H. Chen, L. Xiao, Z. Zhang, and W. Zhu, "Curriculum-graphllm: Joint optimization of architectures, structures and texts for denoised graph neural architecture search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.

[31] X. Wang, L. Xiao, Y. Yao, and W. Zhu, "Ood-graphllm: Graph large language model for out-of-distribution generalized drug synergy prediction," in *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, 2026, pp. 12 315–12 326.

[32] T. Wan, X. Wang, H. Chen, L. Huang, and W. Zhu, "Continual-graphllm: Dynamic graph large language model with invariance regularized adaptive multi-scale experts," in *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, 2026, pp. 4754–4765.

[33] Y. Yao, X. Wang, Y. Meng, Z. Zhang, H. Mei, and W. Zhu, "Disentangled graph llm for molecule graph editing under distribution shifts," in *Proceedings of the ACM Web Conference 2026*, 2026, pp. 1149–1159.

[34] Y. Zhu, H. Shi, X. Wang, Y. Liu, Y. Wang, B. Peng, C. Hong, and S. Tang, "Graphclip: Enhancing transferability in graph foundation models for text-attributed graphs," in *Proceedings of the ACM on Web Conference 2025*, 2025, pp. 2183–2197.

[35] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv *et al.*, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.

[36] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PmLR, 2021, pp. 8748–8763.

[37] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[38] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *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, pp. 3982–3992.

[39] D. Morales-Brotons, T. Vogels, and H. Hendrikx, "Exponential moving average of weights in deep learning: Dynamics and benefits," *Transactions on Machine Learning Research*, 2024.

[40] A. Katharopoulos and F. Fleuret, "Not all samples are created equal: Deep learning with importance sampling," in *International conference on machine learning*. PMLR, 2018, pp. 2525–2534.

[41] M. Paul, S. Ganguli, and G. K. Dziugaite, "Deep learning on a data diet: Finding important examples early in training," *Advances in neural information processing systems*, vol. 34, pp. 20 596–20 607, 2021.

[42] D. Zhao, Y. S. Koh, G. Dobbie, H. Hu, and P. Fournier-Viger, "Symmetric self-paced learning for domain generalization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 15, 2024, pp. 16 961–16 969.

[43] H. Li, X. Wang, and W. Zhu, "Curriculum graph machine learning: A survey," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 2023, pp. 6674–6682.

[44] Y. Qin, X. Wang, Z. Zhang, H. Chen, and W. Zhu, "Multi-task graph neural architecture search with task-aware collaboration and curriculum," *Advances in neural information processing systems*, vol. 36, pp. 24 879–24 891, 2023.

[45] L. Kong, J. Feng, H. Liu, C. Huang, J. Huang, Y. Chen, and M. Zhang, "Gofa: A generative one-for-all model for joint graph language modeling," in *The Thirteenth International Conference on Learning Representations*, 2025.

[46] Z. Wang, Z. Zhang, N. V. Chawla, C. Zhang, and Y. Ye, "Gft: Graph foundation model with transferable tree vocabulary," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. [Online]. Available: https://openreview.net/forum?id=0MXzbAv8xy

[47] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing, "Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality," March 2023. [Online]. Available: https://lmsys.org/blog/2023-03-30-vicuna/

[48] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier *et al.*, "Mistral 7B," *arXiv preprint arXiv:2310.06825*, 2023.

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

[50] X. Sun, H. Cheng, J. Li, B. Liu, and J. Guan, "All in one: Multi-task prompting for graph neural networks," in *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, 2023, pp. 2120–2131.

[51] Y. He, Y. Sui, X. He, and B. Hooi, "Unigraph: Learning a unified cross-domain foundation model for text-attributed graphs," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, 2025, pp. 448–459.

[52] L. Sun, Z. Huang, S. Zhou, Q. Wan, H. Peng, and P. Yu, "Riemanngfm: Learning a graph foundation model from riemannian geometry," in *Proceedings of the ACM on Web Conference 2025*, 2025, pp. 1154–1165.

[53] Z. Wang, Z. Liu, T. Ma, J. Li, Z. Zhang, X. Fu, Y. Li, Z. Yuan, W. Song, Y. Ma *et al.*, "Graph foundation models: A comprehensive survey," *arXiv preprint arXiv:2505.15116*, 2025.

[54] Y. Wang, W. Fan, S. Wang, and Y. Ma, "Towards graph foundation models: A transferability perspective," *arXiv preprint arXiv:2503.09363*, 2025.

[55] J. Liu, C. Yang, Z. Lu, J. Chen, Y. Li, M. Zhang, T. Bai, Y. Fang, L. Sun, P. S. Yu, and C. Shi, "Graph foundation models: Concepts, opportunities and challenges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 6, pp. 5023–5044, 2025.

[56] H. Yuan, Q. Sun, J. Tao, X. Fu, and J. Li, "Rag-gfm: Overcoming in-memory bottlenecks in graph foundation models via retrieval-augmented generation," in *Proceedings of the ACM Web Conference 2026*, 2026, pp. 626–637.

[57] Y. Yan, P. Zhang, Z. Fang, and Q. Long, "Inductive graph alignment prompt: bridging the gap between graph pre-training and inductive fine-tuning from spectral perspective," in *Proceedings of the ACM Web Conference 2024*, 2024, pp. 4328–4339.

[58] X. Yu, Y. Fang, Z. Liu, and X. Zhang, "Hgprompt: Bridging homogeneous and heterogeneous graphs for few-shot prompt learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 38, no. 15, 2024, pp. 16 578–16 586.

[59] X. Yu, C. Zhou, Y. Fang, and X. Zhang, "Multigprompt for multi-task pre-training and prompting on graphs," in *Proceedings of the ACM Web Conference 2024*, 2024, pp. 515–526.

[60] Y. Shen, J. Zhou, B. Bevilacqua, J. Robinson, C. Kanatsoulis, J. Leskovec, and B. Ribeiro, "Zero-shot generalization of GNNs over distinct attribute domains," in *Proceedings of the 42nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 267, 2025, pp. 54 701–54 731.

[61] Y. Fang, D. Fan, S. Ding, N. Liu, and Q. Tan, "Uniglm: Training one unified language model for text-attributed graphs embedding," in *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, 2025, pp. 973–981.

[62] H. Zhao, Z. Li, C. Zi, A. Chen, F. Tsung, J. Li, and J. X. Yu, "A survey of cross-domain graph learning: Progress and future directions," *arXiv preprint arXiv:2503.11086*, 2025.

[63] H. Yuan, Q. Sun, J. Shi, X. Fu, B. Hooi, J. Li, and P. S. Yu, "GRAVER: Generative graph vocabularies for robust graph foundation models fine-tuning," in *Advances in Neural Information Processing Systems*, vol. 38, 2025, pp. 18 738–18 783.

[64] X. Zhu, H. Xue, Z. Zhao, W. Xu, C.-Y. Huang, M. Guo, Q. Wang, K. Zhou, I. Razzak, and Y. Zhang, "LLM as GNN: Graph vocabulary learning for text-attributed graph foundation models," in *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, 2026, pp. 21 373–21 394.

[65] Z. Liu, X. Yu, Y. Fang, and X. Zhang, "Graphprompt: Unifying pre-training and downstream tasks for graph neural networks," in *Proceedings of the ACM web conference 2023*, 2023, pp. 417–428.

[66] X. Jiang, R. Qiu, Y. Xu, W. Zhang, Y. Zhu, R. Zhang, Y. Fang, X. Chu, J. Zhao, and Y. Wang, "Ragraph: A general retrieval-augmented graph learning framework," *Advances in Neural Information Processing Systems*, vol. 37, pp. 29 948–29 985, 2024.

[67] X. Yu, Z. Gong, C. Zhou, Y. Fang, and H. Zhang, "Samgpt: Text-free graph foundation model for multi-domain pre-training and cross-domain adaptation," in *Proceedings of the ACM on Web Conference 2025*, 2025, pp. 1142–1153.

[68] X. Yu, J. Zhang, Y. Fang, and R. Jiang, "Non-homophilic graph pre-training and prompt learning," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, 2025, pp. 1844–1854.

[69] Y. Zhu, X. Li, J. Jia, M. Hu, D. Wu, and M. Qiu, "Towards effective federated graph foundation model via mitigating knowledge entanglement," in *Advances in Neural Information Processing Systems*, vol. 38, 2025, pp. 71 960–71 989.

[70] H. Yuan, Q. Sun, J. Shi, M. Liu, J. Yuan, Z. Zhang, X. Fu, and J. Li, "Retrieving minimal and sufficient reasoning subgraphs with graph foundation models for path-aware graphrag," *arXiv preprint arXiv:2603.07179*, 2026.

[71] H. Yuan, Q. Sun, J. Shi, X. Fu, B. Hooi, J. Li, and P. S. Yu, "How much can transfer? bridge: Bounded multi-domain graph foundation model with generalization guarantees," in *Forty-second International Conference on Machine Learning*, 2025.

[72] P. Ke, H. Ji, Y. Ran, X. Cui, L. Wang, L. Song, X. Zhu, and M. Huang, "Jointgt: Graph-text joint representation learning for text generation from knowledge graphs," in *Findings of the association for computational linguistics: ACL-IJCNLP 2021*, 2021, pp. 2526–2538.

[73] J. Zhao, M. Qu, C. Li, H. Yan, Q. Liu, R. Li, X. Xie, and J. Tang, "Learning on large-scale text-attributed graphs via variational inference," in *The Eleventh International Conference on Learning Representations*, 2023.

[74] H. Chen, X. Wang, Z. Zhang, H. Li, W. Wen, L. Feng, and W. Zhu, "Curriculum GNN-LLM alignment for text-attributed graphs," OpenReview, https://openreview.net/forum?id=tmwR707odU, 2025.

[75] W. Brannon, W. Kang, S. Fulay, H. Jiang, B. Roy, D. Roy, and J. Kabbara, "Congrat: Self-supervised contrastive pretraining for joint graph and text embeddings," in *Proceedings of TextGraphs-17: Graph-based Methods for Natural Language Processing*, 2024, pp. 19–39.

[76] H. Zhang, T. Zhang, Y. Shi, X. Gu, Y. Shen, Z. Zhang, Y. Yuan, H. Zhang, and J. Huang, "Can representation gaps be the key to enhancing robustness in graph-text alignment?" *arXiv preprint arXiv:2510.12087*, 2025.

A Notation

We summarize the main notation used throughout the paper in Table 6.

Table 6: Summary of notations used in this paper.

Notation	Description
$\mathcal{D}$	Set of graph datasets (domains) used in this work.
D	A dataset/domain, $D \in \mathcal{D}$.
$\mathcal{T}$, t_D	Task-type space and the task type of dataset D (node/edge/graph).
$\mathcal{Y}_t$	Label/output space for task type t.
$\{(G_i, y_i)\}_{i=1}^{N_D}$	Labeled graph-task instances in dataset D; N_D is the number of instances.
$G_i = (\mathcal{V}_i, \mathcal{E}_i, \mathbf{X}_i, \mathbf{E}_i)$	i-th graph instance (nodes, edges, node features, edge features).
$\mathcal{V}_i$, $\mathcal{E}_i$	Node set and edge set of G_i.
$\mathbf{X}_i$, $\mathbf{E}_i$	Node features and edge features of G_i.
$\mathrm{GNN}_\phi^{\mathrm{multi}}(\cdot)$, ϕ	Multi-scale GNN encoder and its parameters.
$\mathbf{X}_i^*$	Task-specific graph representation at granularity $* \in \{\text{node, edge, graph}\}$.
$\mathrm{Project}_\theta(\cdot)$, θ	Projector to the LLM embedding space and its parameters.
$\mathbf{z}_i$ (or $\mathbf{z}_i^*$)	Graph token(s) after projection, used as part of the LLM input.
$I_i = [\mathbf{q}_i; \mathbf{z}_i]$	Graph-conditioned instruction sequence.
$\mathbf{q}_i$	Natural-language instruction/query tokens.
$\mathbf{t}_i$	Target output token sequence corresponding to y_i.
$\mathcal{L}$	Training objective (next-token prediction loss).
$\mathcal{D}_{\mathrm{gt}} = \{(G_i, T_i)\}_{i=1}^{N}$	Graph-text pair dataset; T_i is the textual description.
$\mathbf{T}_i$	Text representation encoded from T_i (e.g., Sentence-BERT).
$\mathbf{S}^{g \to t}$	Graph-to-text similarity matrix used in contrastive pretraining.
d_i	Domain index (dataset ID) of instance i.
W_{ab}^g, W_{ab}^t	Domain-aware weights for negative samples (graph/text side).
$\mathcal{B}_k$	Mini-batch at optimization step k.
$\mathcal{D}_k$	Active domains appearing in $\mathcal{B}_k$.
L_i, $L_D^{(k)}$	Per-instance loss and aggregated loss for domain D at step k.
$g_D^{(k)}$	Alignment difficulty score for domain D (gradient norm).
$w_D^{(k)}$	Curriculum weight for domain D at step k.
τ	Temperature in the softmax weighting.

B Algorithm

We provide the complete training pipeline of Uni-GraphLLM in Algorithm 1.

C Instructions Details

C.1 Summarization Prompts

We present the summarization prompts used to generate textual descriptions for graph instances from different graph datasets.

Summarization Prompts Details for Different Graph Datasets

Cora:

I am providing you with a GraphML file depicting a citation network in artificial intelligence research. Each node in the network represents a scholarly article, and each edge signifies a citation relationship between articles. Please analyze the article represented by node 'n{node}' using the provided GraphML data in the following two ways:

1. Paper Summary and Context Analysis:

Algorithm 1 Training Pipeline of Uni-GraphLLM

Input: graph datasets $\mathcal{D} = \{D\}$; graph-text pair dataset $\mathcal{D}_{\text{gt}} = \{(G_i, T_i, d_i)\}_{i=1}^{N}$; instruction-tuning dataset $\mathcal{D}_{\text{it}} = \{(G_i, y_i, d_i)\}$; multi-scale GNN encoder $\text{GNN}_{\phi}^{\text{multi}}$; projector Project_{θ}; text encoder Enc_{text}; warmup ratio ρ; EMA momentum β; temperature τ.
Initialize: GNN parameters ϕ, projector parameters θ, and domain difficulty statistics $\{\mu_D, \tilde{g}_D\}_{D \in \mathcal{D}}$.
Stage I: Graph-Text Pair Pretraining
Compute domain centers and inter-domain distance matrices M_g, M_t according to Eq. (9).
Construct domain-aware weights W^g, W^t according to Eq. (10).
for each mini-batch $\mathcal{B} \subset \mathcal{D}_{\text{gt}}$ **do**
  for each graph-text pair $(G_i, T_i, d_i) \in \mathcal{B}$ **do**
    Extract task-required graph representation $\mathbf{X}_i^*$ with $\text{GNN}_{\phi}^{\text{multi}}$ according to Eqs. (3)–(6).
    Encode text representation $\mathbf{T}_i$ according to Eq. (7).
  end for
  Compute bidirectional similarities $\mathbf{S}^{g \to t}$ and $\mathbf{S}^{t \to g}$ according to Eq. (8).
  Compute the domain-reweighted contrastive loss $\mathcal{L}_{\text{DR-CLIP}}$ according to Eq. (13).
  Update ϕ by minimizing $\mathcal{L}_{\text{DR-CLIP}}$.
end for
Stage II: Curriculum Alignment Tuning
Set total training steps T and warmup horizon $T_w \leftarrow \lfloor \rho T \rfloor$.
for optimization step $k = 1, \ldots, T$ **do**
  Sample mini-batch $\mathcal{B}_k \subset \mathcal{D}_{\text{it}}$ and identify active domains $\mathcal{D}_k$.
  for each sample $(G_i, y_i, d_i) \in \mathcal{B}_k$ **do**
    Extract task-specific graph representation $\mathbf{X}_i^*$ using $\text{GNN}_{\phi}^{\text{multi}}$.
    Project $\mathbf{X}_i^*$ into graph tokens $\mathbf{z}_i^* \leftarrow \text{Project}_{\theta}(\mathbf{X}_i^*)$.
    Form graph-conditioned instruction $I_i = [\mathbf{q}_i; \mathbf{z}_i^*]$ and compute per-instance loss L_i as in Eq. (1).
  end for
  Aggregate dataset-level losses $L_D^{(k)}$ according to Eq. (14).
  Estimate and smooth domain-level alignment difficulty using Eqs. (15) and (16).
  Compute curriculum weights $w_D^{(k)}$ according to Eq. (17).
  Compute curriculum-weighted objective $\mathcal{L}_k$ according to Eq. (18).
  Update θ by minimizing $\mathcal{L}_k$.
end for

- Extract and summarize the key findings or contributions of the paper denoted by 'n{node}'. Consider the details embedded within node 'n{node}', including its title, abstract, and keywords if available.
- Provide an overall summary of prevalent themes or concepts shared by the papers that cite or are cited by 'n{node}', namely its direct neighbors in the network. Identify common threads or research topics among these neighbors.

2. Research Area Classification:
- Based on the information summarized from 'n{node}' and its neighboring nodes, determine the specific research area to which 'n{node}' primarily contributes.
- Justify the classification by explaining which aspects of 'n{node}' align with recognized themes, issues, or methodologies in the identified research area.

Please ensure your analyses are grounded in the data provided by the GraphML file within 400 tokens, focusing on node 'n{node}' and its immediate citation neighborhood. The detailed GraphML citation network data is as follows:
{GraphML}

Pubmed:

I am providing you with a GraphML file depicting a citation network in medical research. Each node in the network represents a scholarly article, and each edge signifies a citation relationship between articles. Please analyze the article represented by node 'n{node}' using the provided GraphML data in the following two ways:

1. Paper Summary and Context Analysis:
- Extract and summarize the key findings or contributions of the paper denoted by 'n{node}'. Consider the details embedded within node 'n{node}', including its title, abstract, and keywords if available.
- Provide an overall summary of prevalent themes or concepts shared by the papers that cite or are cited by 'n{node}', namely its direct neighbors in the network. Identify common threads or research topics among these neighbors.

2. Research Area Classification:
- Based on the information summarized from 'n{node}' and its neighboring nodes, determine the specific research area to which 'n{node}' primarily contributes.
- Justify the classification by explaining which aspects of 'n{node}' align with recognized themes, issues, or methodologies in the identified research area.

Please ensure your analyses are grounded in the data provided by the GraphML file within 400 tokens, focusing on node 'n{node}' and its immediate citation neighborhood. The detailed GraphML citation network data is as follows:
{GraphML}

Arxiv:

I am providing you with a GraphML file depicting a citation network in computer science research from arXiv. Each node in the network represents a scholarly article, and each edge signifies a citation relationship between articles. Please analyze the article represented by node 'n{node}' using the provided GraphML data in the following two ways:

1. Paper Summary and Context Analysis:
- Extract and summarize the key findings or contributions of the paper denoted by 'n{node}'. Consider the details embedded within node 'n{node}', including its title, abstract, and keywords if available.
- Provide an overall summary of prevalent themes or concepts shared by the papers that cite or are cited by 'n{node}', namely its direct neighbors in the network. Identify common threads or research topics among these neighbors.

2. Research Area Classification:
- Based on the information summarized from 'n{node}' and its neighboring nodes, determine the specific research area to which 'n{node}' primarily contributes.
- Justify the classification by explaining which aspects of 'n{node}' align with recognized themes, issues, or methodologies in the identified research area.

Please ensure your analyses are grounded in the data provided by the GraphML file within 400 tokens, focusing on node 'n{node}' and its immediate citation neighborhood. The detailed GraphML citation network data is as follows:
{GraphML}

WikiCS:

I am providing you with a GraphML file that describes a hyperlink-based graph constructed from Wikipedia articles in the field of computer science. Each node represents a Wikipedia entry, and each edge represents a hyperlink relationship between entries.

Please analyze the Wikipedia entry represented by node 'n{node}' based on the provided GraphML data, and complete the following tasks:

1. Content Understanding and Neighborhood Analysis:
- Summarize the main topic and key technical concepts of the Wikipedia entry denoted by 'n{node}', using the entry name and entry content provided in the node.

- Then, examine the immediate neighboring nodes of 'n{node}' in the graph. Identify common themes, technologies, systems, or research topics that appear among these neighboring entries, and describe how they are conceptually related.

2. Topic Classification:
- Based on the content of 'n{node}' and the semantic context provided by its neighboring entries, determine the single most appropriate category for 'n{node}'.
- Provide a brief justification explaining why the selected category best fits the topic and technical focus of 'n{node}'.

Please ensure your analysis is grounded solely in the provided GraphML data, focuses on node 'n{node}' and its immediate neighbors, and is within 400 tokens.
The GraphML data is as follows:
{GraphML}

WN18RR, FB15K237:

I am providing you with a GraphML file that represents a local subgraph from a knowledge graph. Each node corresponds to an entity, and each directed edge represents a semantic relationship between entities.

Please analyze the link specified by the entity pair (head, tail) corresponding to nodes 'n{head}' and 'n{tail}', using the provided GraphML data, and complete the following tasks:

1. Link Semantic Analysis:
- Summarize the semantic meaning of the relationship between the head entity and the tail entity, based on their entity names, descriptions, and the surrounding graph structure. Consider how the head and tail entities are conceptually related.

2. Relation Type Classification:
- Based on your analysis, determine the most appropriate relation type between the head and tail entities.
- Provide a brief justification explaining why this relation best describes the semantic connection between the two entities.

Please ensure that your analysis is grounded solely in the provided GraphML data and focuses on the specified link and its local neighborhood, within 400 tokens.
The GraphML data is as follows:
{GraphML}

ChemHIV, ChemPCBA:

I am providing you with a GraphML file that represents a molecular graph. Each node corresponds to an atom with detailed atomic properties, and each edge represents a chemical bond or structural connection between atoms.

Please analyze the molecular graph using the provided GraphML data and generate a summary by completing the following tasks:

1. Structural Description and Characterization:
- Describe the molecule's overall structure, such as whether it is linear, cyclic, or branched, and its approximate size. Identify and describe the most important functional groups, special bonding patterns, and any notable structural features.

2. Structural Insights for Classification:
- Based on the structural features identified above, infer the most likely chemical property or compound class of this molecule that could be relevant for graph-level classification tasks.
- Provide a brief explanation why these structural characteristics lead to your inference.

Please ensure that your summary is grounded solely in the provided GraphML data and focuses on the overall structure and properties of the entire graph. The summary should be no more than 400 tokens.
The molecular GraphML data is as follows:

{GraphML}

Case of {GraphML}:

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml>
    <key id="k0" for="node" attr.name="title" attr.type="string">
        <default>unkown</default>
    </key>
    <key id="k1" for="node" attr.name="description" attr.type="string"/>
    <key id="k2" for="edge" attr.name="relation" attr.type="string"/>
    <graph id="G" edgedefault="undirected">
        <node id="n2">
            <data key="k0">M.L.(1996) Design by ...</data>
            <data key="k1">Designers rely on past experience ...</data>
        </node>
        <node id="n172">
            <data key="k0">Developing Case-Based Reasoning ...</data>
            <data key="k1">Case-based reasoning ...</data>
        </node>
        <edge id="e0" source="n172" target="n2">
            <data key="k2">cites</data>
        </edge>
        <edge id="e1" source="n2" target="n172">
            <data key="k2">cites</data>
        </edge>
    </graph>
</graphml>
```

Case of Summary:

1. Paper Summary and Context Analysis:
The paper represented by node n2 is titled “M.L. (1996) Design by Interactive Exploration Using Memory-Based Techniques.” It discusses the use of memory-based techniques from artificial intelligence to aid in the design process, particularly in the domain of structural design. The paper introduces DEMEX, an interactive computer-aided design system that leverages memory-based techniques to help users explore and refine design problems. The key contribution is the integration of memory-based methods to support exploration and understanding of ill-structured design problems.
The paper’s neighbors include n172, titled “Developing Case-Based Reasoning for Structural Design.” This paper explores the development of case-based reasoning systems for design, including the implementation of four systems: CASECAD, CADSYN, WIN, and DEMEX. It emphasizes the use of case memory organization, recall, and adaptation in design tasks.
The shared themes between n2 and n172 include: - Memory-based techniques for design. - Case-based reasoning as a method for supporting design tasks. - Interactive exploration in the early stages of design. - Structural design as the application domain. - Adaptation of design cases to fit new contexts.
These themes suggest a strong focus on case-based reasoning and interactive design exploration in the context of structural design.
2. Research Area Classification:
Based on the analysis, node n2 primarily contributes to the research area “Case_Based”.
Justification: - The paper introduces DEMEX, a system that uses memory-based techniques to support design exploration, which aligns with case-based reasoning. - The paper is cited by and cites n172, which explicitly discusses the development of case-based reasoning systems for structural design. - The shared themes of case memory organization, case recall, and case adaptation are central to case-based reasoning. - The focus on interactive exploration and design problem refinement is consistent with the broader goals of case-based reasoning in design.

Thus, n2 is best classified under "Case_Based" due to its direct contribution to the development and application of case-based reasoning techniques in design.

C.2 Instruction Tuning Prompts

We present the instruction tuning prompts used to align graph representations with LLMs across different graph datasets. Notably, for *FB15K237*, including all 237 relation candidate labels in the instruction would result in excessively long input sequences for all GLMs. To reduce the token length, we adopt a 10-way sampling strategy: each instruction includes 10 candidate labels, consisting of the correct label and 9 negative labels randomly sampled from the remaining labels.

Instruction Tuning Prompts Details for Different Graph Datasets

Cora:

Given a representation of a paper: <Node 1>, with the following information:
Title: {title}
Abstract: {abstract}
Question: Which machine learning sub-category does this paper belong to? Please directly give the most likely answer from the following sub-categories: {candidate_labels}.

PubMed:

Given a representation of a paper: <Node 1>, with the following information:
Title: {title}
Abstract: {abstract}
Question: Which Medical Subject Headings (MeSH) term does this paper belong to? Please directly give the most likely answer from the following MeSH terms: {candidate_labels}.

Arxiv:

Given a representation of a paper: <Node 1>, with the following information:
Title: {title}
Abstract: {abstract}
Question: Which arXiv CS sub-category does this paper belong to? Please directly give the most likely answer from the following sub-categories: {candidate_labels}.

WikiCS:

Given a representation of a Wikipedia page: <Node 1>, with the following information:
Name: {name}.
Content: {content}.
Question: Which category does this Wikipedia page belong to? Please directly give the most likely answer from the following categories: {candidate_labels}.

WN18RR, FB15K237:

Given a representation of relation between two entities: <Node 1>, with the following information:
First entity: Name: {head_name}, Description: {head_description}.
Second entity: Name: {tail_name}, Description: {tail_description}.
Question: Which category should the relation between these two entities be classified as? Please directly give the most likely answer from the following categories: {candidate_labels}.

ChemHIV, ChemPCBA:

Given a representation of a molecule <Node 1>, with the following information:

SMILES: {smiles}.
Question: {task} Please answer: "Yes, this molecule is effective to this assay" or "No, this molecule is not effective to this assay".

D Experiment Details and Additional Results

D.1 Datasets Details

Dataset Statistics. We adopt the preprocessing pipeline of [46], where raw textual descriptions associated with nodes and edges are encoded into 768-dimensional representations using a Sentence-BERT [38]. For knowledge graphs (KGs), edge textual information is not further encoded as features, since the original textual content is already sufficient for KG completion tasks. Detailed dataset statistics are summarized in Table 7 and describe the dataset details as follows.

Table 7: Dataset statistics. The number of graph instances depends on the task type: for node classification, it corresponds to the number of nodes; for edge classification, it corresponds to the number of edges; and for graph classification, it equals the number of graphs.

Dataset	Task	# Graphs	Avg. # Nodes	Avg. # Edges	# Graph Instances	# Classes
Cora	Node	1	2,708	10,556	2,708	7
PubMed	Node	1	19,717	44,338	19,717	3
Arxiv	Node	1	169,343	1,166,243	169,343	40
Wiki-CS	Node	1	11,701	216,123	11,701	10
FB15K237	Link	1	14,541	310,116	310,116	237
WN18RR	Link	1	40,943	93,003	93,003	11
ChemPCBA	Graph	437,929	26.0	28.1	437,929	128
ChemHIV	Graph	41,127	25.5	27.5	41,127	2

Dataset Splitting. We follow the same data splitting strategy as [11]. For datasets with multiple predefined splits, we adopt the first split for consistency. During the graph-text pair pretraining stage, we utilize all available data to train the GNN encoder. In the instruction tuning stage, we train on the training split of each dataset. For large-scale datasets such as *Arxiv* and *FB15K237*, we randomly sample subsets for training (Arxiv: 40,000; FB15K237: 30,000), while for smaller datasets, we use the full training set. Notably, the training sets of *ChemPCBA* and *ChemHIV* exhibit severe class imbalance, which can degrade the performance of all methods. Since handling class imbalance is not the primary focus of this work or GLM-based approaches, we apply balanced sampling subset from the training set to ensure the model can learn meaningful signals. For datasets with a limited number of graph instances, we increase the training data through repeated sampling, using larger effective training sizes (Cora: 20×, PubMed: 20×, ChemHIV: 10×). All methods use the aforementioned training data for instruction tuning. During evaluation, we report performance on the test split of each dataset.

Evaluation Metrics. For node- and edge-level classification, we report classification accuracy (ACC%) and F1-score (F1%), for graph-level tasks on molecular datasets, we report the area under the ROC curve (AUC%).

D.2 Baselines and Implementation Details

We compare Uni-GraphLLM with a range of baselines, including pure LLMs and state-of-the-art GLMs. For a fair comparison, we use the same LLM backbone, Vicuna-7B-v1.5 [47], for our model and all baselines except GOFA, which uses Mistral-7B-Instruct-v0.2 [48]. Under the multi-domain, multi-task learning setting, for baselines that require retraining, we use pretraining data of the same scale and apply the same repeated sampling strategy as described in Appendix D.1. Under the cross-domain and cross-task generalization settings, GraphGPT, GOFA, and LLaGA are trained on the full Arxiv training set, whereas TEA-GLM and Uni-GraphLLM are trained on 40,000 graph instances sampled from the Arxiv training set.

- **LLM Baselines.** We evaluate the performance of pure LLMs without any graph-specific encoding or alignment, including Vicuna-7B-v1.5 [47] and LLaMA-2-7B [49]. These models are directly prompted with natural language descriptions of the graph tasks without any additional graph representation.
- **GraphGPT [8].** GraphGPT is a graph-oriented instruction-tuned framework that augments a pre-trained LLM with graph structural representations. It aligns graph structural information with the natural language space through text-graph grounding, and maps graph embeddings into LLM-compatible graph tokens via a lightweight alignment projector within a two-stage graph instruction tuning paradigm. Since its released training and evaluation pipelines are primarily designed for node-level presentations, we pretrain GraphGPT on all node-level datasets and restrict its evaluation to these datasets.
- **LLaGA [29].** LLaGA adapts graph data into structure-aware node sequences that preserve structural information, which are then projected into the LLM token embedding space via a projector. Since the original implementation of LLaGA only supports node-level presentations, we pretrain it on all node-level datasets and restrict evaluation to these datasets.
- **GOFA [45].** GOFA proposes a generative graph language model by interleaving GNN layers with a pre-trained LLM, enabling joint modeling of graph structure and textual semantics. It is pre-trained with a unified generative objective over multiple graph-related tasks, such as graph-level prediction and question answering. Due to the substantial computational cost of pretraining GOFA from scratch, we directly adopt the released checkpoints from the original paper for evaluation under the multi-domain, multi-task learning setting and convert the datasets used in our paper into the input format required by GOFA.
- **TEA-GLM [10].** TEA-GLM introduces a text-enhanced graph alignment framework that aligns GNN-encoded graph representations with LLM token embeddings via instruction tuning, leveraging textual information to bridge graph and language modalities. The original implementation only supports node-level presentations. We extend it in a principled manner to support edge-level and graph-level tasks, and then retrain its GNN encoder and projector on multi-task, multi-domain data so that evaluation can be carried out across all task types.

D.3 Complete Main Results

We present the complete main results as in Table 8, including accuracy and F1-score for node- and edge-level tasks, as well as AUC for graph-level tasks.

Table 8: Complete performance comparison of different methods on node classification, edge classification, and graph classification tasks under multi-domain and multi-task learning. The highest result is **bold**, and the second highest result is underlined.

Method	Node Classification								Edge Classification				Graph Classification	
	Cora		PubMed		Wiki-CS		Arxiv		WN18RR		FB15K237		ChemPCBA	ChemHIV
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	AUC	AUC
Vicuna-7B	56.67	58.52	78.92	78.85	62.34	60.57	43.75	23.08	34.84	12.14	56.01	38.46	46.47	30.36
LLaMA-2-7B	29.45	30.13	25.36	19.66	22.22	21.46	6.33	3.40	29.16	7.54	20.60	11.30	51.12	50.09
GraphGPT	57.59	57.91	73.45	72.85	49.20	36.87	67.63	34.73	N.S.	N.S.	N.S.	N.S.	N.S.	N.S.
LLaGA	74.71	73.29	68.54	67.96	69.16	63.08	73.83	55.86	N.S.	N.S.	N.S.	N.S.	N.S.	N.S.
GOFA	63.88	60.88	64.72	48.46	66.77	62.85	66.11	46.16	21.95	10.00	54.41	33.94	60.61	49.78
TEA-GLM	62.38	59.71	85.13	84.32	55.19	50.27	66.81	44.03	92.85	73.82	91.85	**75.21**	80.40	67.13
Ours	**76.98**	**75.72**	**90.49**	**90.09**	**76.65**	**72.40**	**74.22**	**56.03**	**93.52**	**75.30**	**91.96**	74.00	**81.69**	**70.09**

D.4 Efficiency Comparison with Baselines

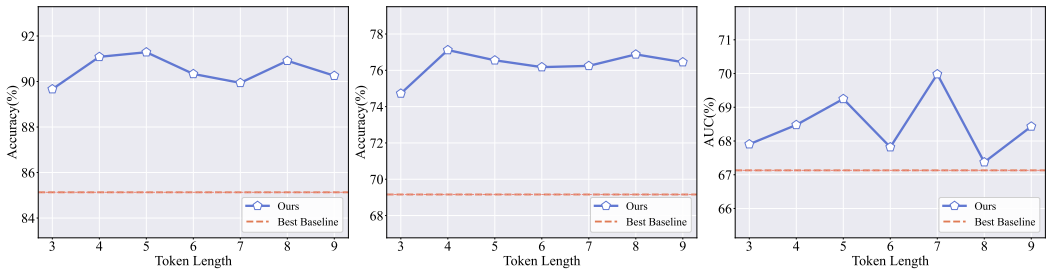
We compare the time and memory consumption of Uni-GraphLLM with representative baselines under the cross-domain and cross-task generalization settings in Table 9. We retain the original batch size of each baseline whenever possible. GraphGPT, LLaGA, and GOFA are trained on the full Arxiv training set, whereas TEA-GLM and Uni-GraphLLM are trained on 40,000 graph instances sampled from the Arxiv training set.

Table 9: Efficiency comparison of different methods across datasets, including runtime, batch size, and GPU memory usage.

	Arxiv (Training)			**Cora (Inference)**			**PubMed (Inference)**		
Method	Time (s)	Batch size	Mem (MB)	Time (s)	Batch size	Mem (MB)	Time (s)	Batch size	Mem (MB)
GraphGPT	22641	1	30717	338	1	14550	8200	1	14651
LLaGA	18286	16	34447	3339	1	13222	19260	1	13215
GOFA	137175	1	53100	3385	1	22732	39961	1	23083
TEA-GLM	4866	2	19941	327	20	16142	1075	20	16501
Ours	11072	3	34953	181	20	30891	2136	20	32250

D.5 Hyperparameters Analysis

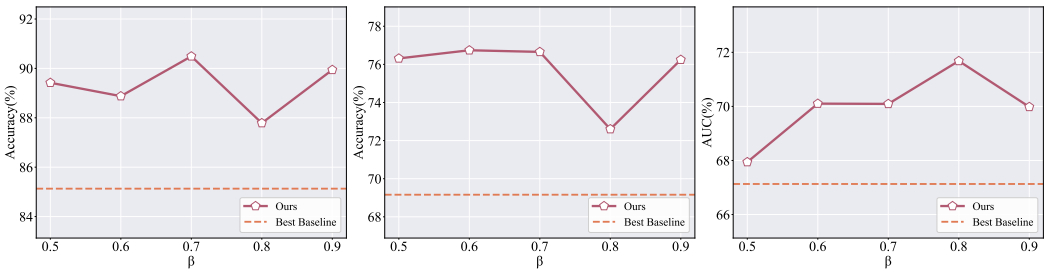
Graph Token Length. We study the sensitivity of Uni-GraphLLM to the number of graph tokens m injected into the LLM. Intuitively, a larger m provides a higher-capacity interface for conveying fine-grained graph information (e.g., more queried nodes/edges or richer multi-scale features) to the LLM, but also increases the compute and memory cost due to the quadratic attention complexity in the input length (Sec. D.6). We vary m in a moderate range (e.g., $m \in \{3, 4, 5, 6, 7, 8, 9\}$) while keeping other settings fixed, and report the resulting accuracy or AUC on several datasets. As shown in Figure 3, performance generally improves as m grows from very small values and then saturates, with overly large m yielding diminishing performance. Across the tested range, Uni-GraphLLM consistently outperforms the best baseline, indicating that its advantage is robust to the choice of graph token length.



(a) PubMed (b) Wiki-CS (c) ChemHIV

Figure 3: Hyperparameter analysis of the graph token length m. The solid line denotes the performance of Uni-GraphLLM, while the dashed line indicates the best-performing baseline.

Curriculum EMA Momentum. We further analyze the sensitivity of the curriculum alignment tuning to the EMA momentum β in Eq. (16), which controls the smoothness of the domain difficulty estimate $\tilde{g}_D^{(k)}$. A small β makes the estimate highly responsive but potentially noisy, leading to unstable domain reweighting; a large β yields a smoother curriculum but may react slowly to shifts in optimization dynamics, especially during early training. We vary β over a broad range (e.g., $\beta \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$) and report the final performance in Figure 4. Uni-GraphLLM remains stable across a wide range of β and consistently outperforms the best baseline, suggesting that the curriculum mechanism is not overly sensitive to this hyperparameter.



(a) PubMed (b) Wiki-CS (c) ChemHIV

Figure 4: Hyperparameter analysis of the EMA momentum β. The solid line denotes the performance of Uni-GraphLLM, while the dashed line indicates the best-performing baseline.

D.6 Time Complexity Analysis

We analyze the per-instance time complexity for a single graph-conditioned input (G, I), where $G = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E})$ denotes the input graph and I is the corresponding instruction with text length L_t. For a message-passing GNN with L_g layers and hidden dimension d_g, each layer incurs $\mathcal{O}(|\mathcal{E}|d_g)$ for neighborhood aggregation and $\mathcal{O}(|\mathcal{V}|d_g^2)$ for node-wise transformations, resulting in an overall complexity of $\mathcal{O}\big(L_g(|\mathcal{E}|d_g + |\mathcal{V}|d_g^2)\big)$ for computing node representations. A global pooling operation further adds $\mathcal{O}(|\mathcal{V}|d_g)$ to obtain a graph-level representation. Task-specific representations (Eqs. 4–6) are then constructed via a $2d_g \rightarrow d_g$ linear transformation over concatenated graph features, incurring a cost of $\mathcal{O}(d_g^2)$. These representations are subsequently projected into m graph tokens in the LLM embedding space of dimension d_l, which requires $\mathcal{O}(md_gd_l)$. Finally, given a text sequence of length L_t augmented with m graph tokens, a Transformer LLM with L_l layers has a dominant self-attention complexity of $\mathcal{O}\big(L_l(L_t + m)^2 d_l\big)$. Overall, the end-to-end time complexity is $\mathcal{O}\big(L_g(|\mathcal{E}|d_g + |\mathcal{V}|d_g^2) + md_gd_l + L_l(L_t + m)^2 d_l\big)$.

D.7 Implementation Details

During graph-text pair pretraining, we use GraphSAGE [37] as the GNN backbone, with 3 layers and 768 for the input, hidden, and output dimensions. The multi-scale GNN encoder is trained for 100 epochs with a batch size of 4096 and a learning rate of 1×10^{-4}. For instruction tuning, we initialize the GNN encoder from the pretrained checkpoint and use Vicuna-7B-v1.5 [47] as the default LLM backbone. The LLM backbone is frozen, and the alignment module is trained for one epoch with a batch size of 3 and a learning rate of 0.004. We set the graph token length m to 7. For curriculum alignment tuning, we set the warmup ratio ρ to 0.01 and the EMA momentum β to 0.7. For the backbone robustness experiment, we replace Vicuna with Qwen3-8B [35], adapt the projector output dimension and instruction template, and otherwise retain the same framework and multi-domain, multi-task setting. We pretrain the GNN encoder on 6 NVIDIA GeForce RTX 3090 GPUs, each with 24GB of memory, and perform instruction tuning on NVIDIA A100-SXM4-80GB GPUs.

E More Related Works

Multi-domain Multi-task Graph Learning. Recent mainstream efforts in multi-domain and multi-task graph learning have increasingly moved toward Graph Foundation Models (GFMs) [50, 51, 52, 53, 54, 55, 56], which aim to develop graph models that can generalize beyond a single dataset, domain, or task setting [57, 58, 59, 60, 61, 62, 63, 64]. Instead of training separate models for each graph domain or downstream task, GFMs typically seek to share model parameters, representation spaces, prompting mechanisms, or pretraining objectives across heterogeneous graph data, thereby improving generalization and transferability [65, 66, 67, 68, 69, 70, 71]. Representative works have explored how to unify graph inputs, task formats, and transferable structural patterns across domains. For example, OFA [11] formulates diverse graph datasets as text-attributed graphs, where nodes and edges are described using natural language and encoded into a shared feature space. It further introduces nodes-of-interest and graph prompting mechanisms to standardize different

graph classification tasks under a unified model. GFT [46], on the other hand, studies transferable graph patterns from the perspective of message passing by treating computation trees as reusable vocabulary tokens, enabling cross-domain and cross-task transfer through a shared tree vocabulary. AutoGFM further customizes GNN architectures across domains and tasks and uses curriculum-based architecture customization to mitigate data domination during multi-domain optimization [13]. Nevertheless, most existing methods focus on improving the generalization ability of graph encoders themselves, without explicitly studying how multi-domain, multi-task GNN representations should be aligned with LLMs to construct a graph language model. In contrast, our work targets this alignment problem by learning a shared GNN encoder from diverse graph-text pairs and adaptively aligning its representations with LLMs across domains and tasks.

Graph-Text Alignment. Graph-text alignment has been widely explored as an effective way to bridge graph-structured data and natural language semantics [72, 73, 74]. A common strategy is to construct paired graph-text views, such as node-text, graph-text, or substructure-text pairs, and jointly train graph and text encoders with contrastive or matching objectives, so that structurally meaningful graph representations can be aligned with textual semantics [75, 34, 76]. Such alignment is particularly useful for text-attributed graphs and graph-language applications, where textual attributes provide rich semantic supervision beyond graph topology and task labels. Representative methods mainly study graph-text alignment from the perspective of self-supervised representation learning. For example, ConGraT [75] proposes a general contrastive graph-text pretraining framework for text-attributed graphs, where a language model and a graph neural network are trained to align text and node representations in a shared latent space through a CLIP-style contrastive objective. GraphCLIP [34] enhances the transferability of graph foundation models for text-attributed graphs by constructing LLM-generated graph-summary pairs and aligning graph and summary representations through self-supervised contrastive pretraining with invariant learning. However, existing graph-text alignment methods are typically designed for specific domains and tasks, and do not explicitly consider how to learn generalizable graph representations that can be aligned with LLMs across diverse domains and tasks. Our approach differs by using graph-text pair pretraining to obtain generalizable, text-aligned GNN representations and further performing curriculum alignment tuning to connect these representations with the LLM token space.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the paper's scope and the claims are supported by the experimental results on multi-domain/multi-task learning, cross-domain generalization, and cross-task generalization.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the limitations of the work in Conclusion section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not present theoretical theorems or formal proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper describes the model components, training objectives, full training algorithm, datasets, splits, baselines, prompts, evaluation metrics, and additional implementation details in the main text and appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All datasets used in this paper are publicly available, and our code is released at `https://github.com/haibo12/Uni-GraphLLM` with instructions for reproducing the main results. Experimental details are provided in Section 5 and Appendix D.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: The experimental setup describes data splits, hyperparameters and implementation details in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The paper does not report error bars as repeated runs would incur substantial computational cost.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: This paper reports runtime, batch size, and GPU memory usage for representative training and inference settings, and provides a time-complexity analysis.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses public graph datasets and pretrained models for methodological research, and we are not aware of any deviation from the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The paper focuses on foundational graph-language modeling methods and is not directly tied to real-world deployment or societal applications; therefore, we do not anticipate immediate societal impacts.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk pretrained language models, image generators, or scraped datasets; it builds on existing public graph datasets and pretrained LLM backbones.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the datasets, baseline methods, and pretrained models used in the experiments and respects the licenses and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release the code of Uni-GraphLLM at `https://github.com/haibo12/Uni-GraphLLM`. The repository documents the environment, data preparation, and usage, together with its license.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

Justification: The paper does not involve crowdsourcing experiments or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The paper describes the use of Qwen3-8B to generate textual descriptions for graph instances and specifies the LLM backbones used for the proposed method and baselines.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.