

Scientific Report: Summary of 54 Research Articles on Language Models in Graph-Based Systems

1.

R. Takeda, H. Munakata, and K. Komatani, “Link prediction based on large language model and knowledge graph retrieval under open-world and resource-restricted environment,” ””, 2023.

Title: *Link Prediction LLM and Knowledge Graph Retrieval under Open-World and Resource-Restricted Environment*

This paper addresses the task of predicting the tail entity in a knowledge graph, where the type of the tail entity aligns with the relation type. In open-world knowledge graphs, some entities may not be present in the graph itself; therefore, large language models (LLMs) can be pre-trained to generate textual representations for such unknown entities. Unlike classification problems, where the goal is to determine the existence of triples, link prediction focuses on identifying the appropriate tail entity. The overall process proposed in this paper begins with a knowledge graph (KG) retrieval step, where example facts are extracted using algorithms such as FewerHopsEx and RandEx—both of which act as sampling mechanisms. A prompt is then constructed using one example (+ relation: for $i = n$) along with a target query (head + relation: ?), and this prompt is fed into the LLM. Following the Monte Carlo approach, the LLM is sampled 3,000 times to generate candidate noun phrases. Finally, each noun phrase is scored based on the probability assigned by the LLM and its posterior probability, enabling the ranking and selection of the most suitable tail entity.

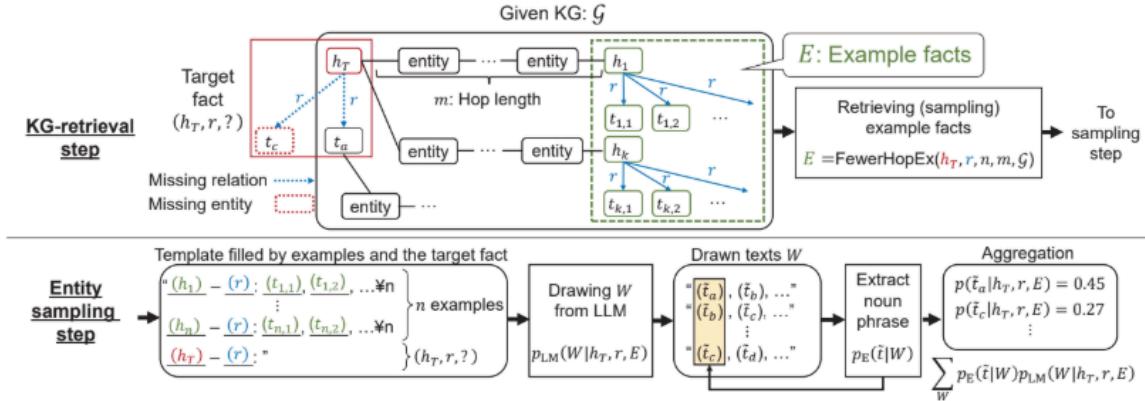


Figure 2: Overview of proposed method. Target fact $(h_T, r, ?)$ and n examples $\{(h_1, r, t_{1,1}), (h_1, r, t_{1,2}, \dots)\}, \dots, \{(h_n, r, t_{n,1}), (h_n, r, t_{n,2}, \dots)\}$, related to h_T , are retrieved from KG by our algorithm. Obtained facts are then incorporated into a template (lower left) from which output texts are repeatedly generated by LLM. Finally, the first noun phrase is extracted as predicted fact $(\tilde{t}_a, \tilde{t}_b, \dots)$ and aggregated as $p(\tilde{t}|h_T, r, E)$.

score. We assume the posterior probability is represented as the following marginalized probability:

$$p(\tilde{t}|h_T, r, \mathcal{G}) = \sum_W p_E(\tilde{t}|W, \mathcal{G})p_{LM}(W|h_T, r, \mathcal{G}), \quad (1)$$

Table 4: Examples of a prompt, predicted tail entities and their score with FewerHopEx, and true tail entities. Italic font indicates English translations.

Prompt	Predicted tail entity: Score	True tail entity
チキンライス - 材料: ケチャップ, 米, 鶏肉 \n ロールキャベツ - 材料: <i>chicken rice - made from material:</i> <i>ketchup, rice, chicken</i> \n <i>cabbage roll - made from material:</i>	キャベツ (<i>cabbage</i>): 0.679, 玉ねぎ (<i>onion</i>): 0.056, 鶏肉 (<i>chicken</i>): 0.051, 野菜 (<i>vegetable</i>): 0.040, じゃがいも (<i>potato</i>): 0.027, 米 (<i>rice</i>): 0.023, ⋮	キャベツの葉 (<i>cabbage leaves</i>). 玉ねぎ (<i>onion</i>), 豚ひき肉 (<i>minced pork</i>) ひき肉 (<i>minced meat</i>) 肉 (<i>meat</i>), コンソメ (<i>consommé</i>)
乾パン - 同一とされる事物: ツヴィーパック \n マルゲリータ - 同一とされる事物: hardtack - said to be the same as: zwieback \n Margherita - said to be the same as:	ピザ (<i>pizza</i>): 0.380, ピツツア (<i>pizza</i>): 0.072, パン (<i>bread</i>): 0.044, ツヴィーパック (<i>zwieback</i>): 0.037, ミケーレ (<i>Michele</i>): 0.027, ミケランジェロ (<i>Michelangelo</i>): 0.019, ⋮	ナポリピツツア (<i>Neapolitan pizza</i>)

2.

R. Ong, J. Sun, Y.-K. Guo, and O. Serban, “Dynamic link prediction: Using language models and graph structures for temporal knowledge graph completion with emerging entities and relations,” *Expert Systems with Applications*, vol. 272, p. 126648, 2025. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0957417425002702>

Title: *Dynamic Link Prediction: Using Language Models and Graph Structures for Temporal KGC with Emerging Entities*

In temporal knowledge graph completion, multiple snapshots of the knowledge graph (KG) are considered over time. Each fact is represented as a triple (head, relation, tail), which can be either seen (present in training) or unseen. The representation of an entity is derived from its one-hop neighboring relations. In this study, the proposed method for KG link prediction utilizes the RoBERTa language model to encode the textual representation of relations into relation embeddings, while also incorporating the graph structure. These encodings are then combined and used in a traditional scoring function for link prediction, specifically the TransE score function. The study employs three types of entity encodings: (1) RoBERTa-based encoding of the entity's textual description, (2) factual relation embeddings, and (3) embeddings of neighboring relations surrounding the target entity, including both neighboring head and tail relations. These components are integrated to construct comprehensive entity representations for improved link prediction.

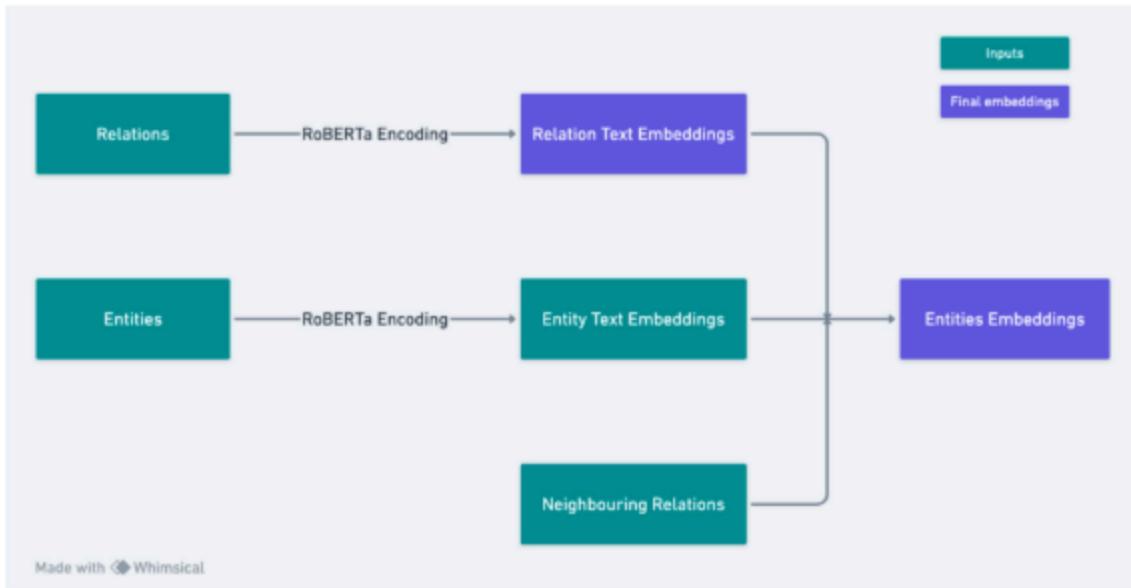


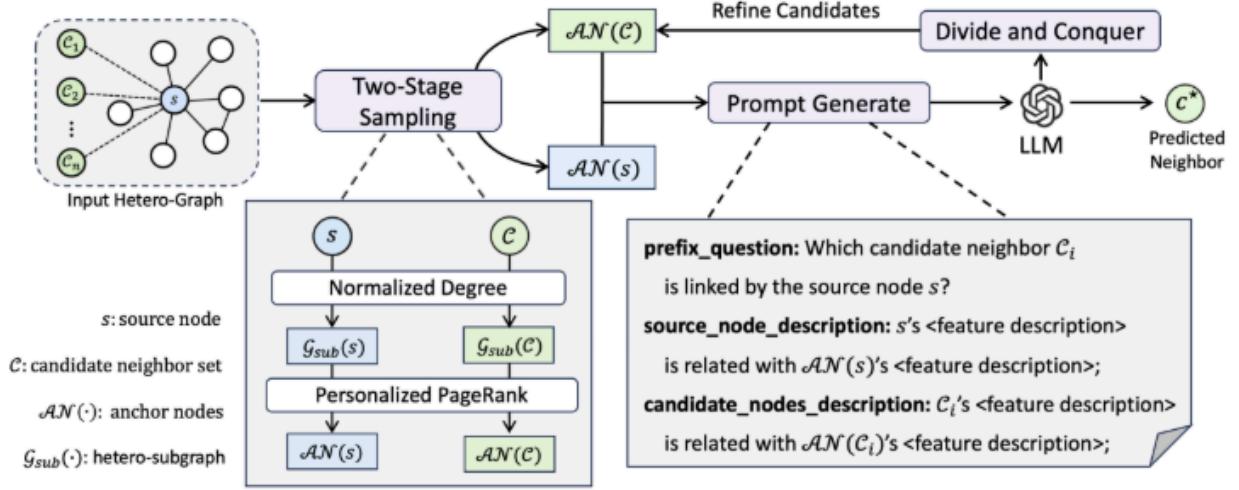
Fig. 1. Our proposed methodology in encoding seen and unseen entities and relations.

3.

B. Bi, S. Liu, Y. Wang, L. Mei, and X. Cheng, “LpnL: Scalable link prediction with large language models,” 2024. [Online]. Available: <https://arxiv.org/abs/2401.13227>

Title: LPNL: Scalable Link Prediction with LLMs

This study focuses on link prediction in heterogeneous graphs using large language models (LLMs). The proposed prompt for each node, denoted as $d(v)$, consists of a sequence of textual features from the node itself and its k -hop anchor nodes. Two major challenges are addressed: (1) prompt length inflation due to high-hop neighborhoods and (2) degree variability across different node types. To tackle these issues, a two-stage sampling strategy is employed. First, subgraphs are sampled based on normalized node degrees. Second, top- k anchor nodes are selected using Personalized PageRank (PPR) to construct $d(v)$. To mitigate the token overhead problem, the authors apply a divide-and-conquer strategy by randomly splitting candidate nodes into fixed-length sets. Additionally, they use self-supervised fine-tuning of the LLM by masking portions of the input content, enhancing the model's ability to generalize in link prediction tasks.



central node s , the sampling probability of node v can be described as follows:

$$prob_s^l(v) = \frac{\deg(v)^2}{\sum \deg(u)^2} \quad (3)$$

where node u represents the neighbor node at the l -th layer within the subgraph, satisfying $u \in \mathcal{V}_s^h \setminus \mathcal{V}_s^{h-1}$ and $\phi(v) = \phi(u)$.

First, we define $d(v)$ as the description of node v , which consists of a sequence of textual features of itself and also its top- k anchor nodes:

$$d(v) = \{v : \mathcal{S}_v\} \text{ is related with } \sum_{i=1}^k \{v'_i : \mathcal{S}_{v'_i}\} \quad (1)$$

where \mathcal{S}_v denotes the textual description of node v and v'_i represents the anchor node of node v satisfying $v'_i \in \mathcal{AN}_k^h(v)$.

Subsequently, given a source node s and the set of candidate neighbors \mathcal{C} , we formally obtain the link prediction prompt template as follows:

$$\mathcal{T}(s, \mathcal{R}, \mathcal{C}) = q(\mathcal{R}) + d(s) + \sum_{i=1}^n d(c_i | c_i \in \mathcal{C}) \quad (2)$$

Author Disambiguation Example

prefix_question: Which following candidate author writes the paper p₁?

source_node_description: p₁: <paper title> is related with f₂₅: <field name>, v₁₃: <journal info>, p₄₆: <paper title>, a₃₈: <author info>, p₂₇: <paper title>...

candidate_nodes_description: a₁: <author info> is related with p₁₅: <paper title>...; a₂: ...; a₃: ...

D. Shu, T. Chen, M. Jin, C. Zhang, M. Du, and Y. Zhang, “Knowledge graph large language model (kg-llm) for link prediction,” 2024. [Online]. Available: <https://arxiv.org/abs/2403.07311>

Title: *Knowledge Graph Large Language Model (KG-LLM) for Link Prediction*

In this study, the authors depart from traditional graph embedding techniques and graph neural networks (GNNs) by leveraging instruction fine-tuning (IFT) on large language models (LLMs), converting knowledge graph (KG) paths into natural language prompts. The study addresses two tasks: (1) multi-hop link prediction, which determines whether a path exists between two nodes across multiple hops, and (2) multi-hop relation prediction, which predicts the relation between the first and last nodes by identifying a connecting path of triples. To support these tasks, a preprocessing step uses a depth-first search (DFS) algorithm to extract paths from the KG, ranging from 2 to 6 nodes, and labels them as positive (if a link exists between the first and last nodes) or negative. The study introduces a prompt construction strategy to convert these paths into natural language, with two types of prompts: Ablation prompts (lacking explicit instructions or reasoning) and KG-LLM prompts (which include detailed instructions and chain-of-thought (CoT) reasoning). During inference, the method utilizes in-context learning (ICL), providing an example prompt to enable the model to generalize better to unseen prompt types.

Multi-hop Link Prediction (Ablation)	Multi-hop Link Prediction (KG-LLM)	Multi-hop Relation Prediction (Ablation)	Multi-hop Relation Prediction (KG-LLM)
<div style="border: 1px solid #ccc; padding: 10px;"> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 5px;">PROMPT</div> <pre>### Input: Node [node id1] has relation [relation id] with node [node id2]. Node [node id2] has relation [relation id] with node [node id3]. [...]</pre> <div style="border: 1px solid #ccc; padding: 2px; margin-top: 10px;">Expected Output</div> <pre>### Response: [Yes / No]</pre> </div>	<div style="border: 1px solid #ccc; padding: 10px;"> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 5px;">PROMPT</div> <pre>### Instruction: Below is the detail of a knowledge graph path. Is node [node id1] connected with node [last node id]? Answer the question by reasoning step-by-step. Choose from the given options: 1. Yes 2. No</pre> <div style="border: 1px solid #ccc; padding: 2px; margin-top: 10px;">Expected Output</div> <pre>### Response: Node [node id1] has relation [relation id] with node [node id2]. Node [node id2] has relation [relation id] with node [node id3]. [...]</pre> </div>	<div style="border: 1px solid #ccc; padding: 10px;"> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 5px;">PROMPT</div> <pre>### Input: Node [node id1] has relation [relation id] with node [node id2]. Node [node id2] has relation [relation id] with node [node id3]. [...]</pre> <div style="border: 1px solid #ccc; padding: 2px; margin-top: 10px;">Expected Output</div> <pre>### Response: [relation id]</pre> </div>	<div style="border: 1px solid #ccc; padding: 10px;"> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 5px;">PROMPT</div> <pre>### Instruction: Below is the detail of a knowledge graph path. What is the relation between [node id1] and [last node id]? Answer the question by reasoning step-by-step. Choose from the given options: 1. [relation text1] 2. [relation text2] [...]</pre> <div style="border: 1px solid #ccc; padding: 2px; margin-top: 10px;">Expected Output</div> <pre>### Response: Node [node id1] has relation [relation id] with node [node id2]. Node [node id2] has relation [relation id] with node [node id3]. [...]</pre> </div>

Z. He, J. Zhu, S. Qian, J. Chai, and D. Koutra, “Linkgpt: Teaching large language models to predict missing links,” 2024. [Online]. Available: <https://arxiv.org/abs/2406.04640>

Title: *LinkGPT: Teaching LLMs to Predict Missing Links*

This study focuses on homogeneous text-attributed graphs (TAGs) and aims to integrate both node-wise and pair-wise structural information into a large language model (LLM). One of the key challenges addressed is the difficulty of teaching LLMs to predict links effectively. To support this, two special tokens are introduced: <NODE>, which encodes information about a node and its neighbors, and <PAIRWISE>, which captures the pairwise relationship between nodes. The neighborhood-aware node encoding combines the textual representation of a node's neighborhood with the node's own representation, and is optimized using contrastive loss. For pairwise encoding, the study employs a model called LPFormer. Instruction tuning is performed in two stages: Stage 1 involves aligning the pairwise encoder and the special token encodings with the word embeddings of a pre-trained LLM; Stage 2 involves fine-tuning the LLM itself using Low-Rank Adaptation (LoRA). Additionally, a retrieval stage is implemented to identify a small subset of candidate nodes using a neighbor generation prompt to the LLM, followed by an algorithmic filtering of the generated candidates. Finally, a reranking stage leverages the LLM to assign probabilities and re-rank the final candidate links.

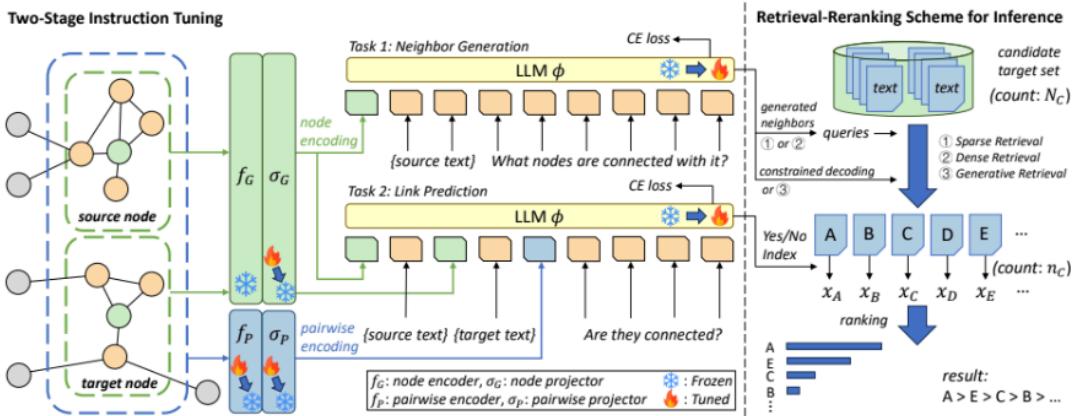
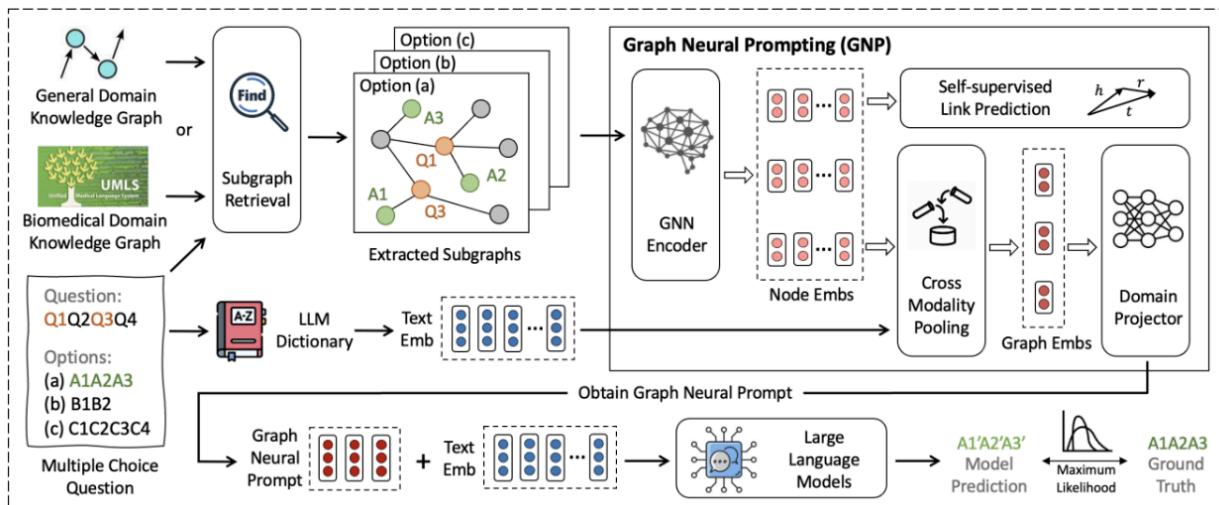


Figure 1: Overview of the LINKGPT framework. LINKGPT consists of two main components: (1) two-stage instruction tuning approach to incorporate structural information into LLMs, and (2) a retrieval-reranking scheme to address the computation bottleneck at inference time. Note that, $\heartsuit \Rightarrow$ means that this module is tuned during stage 1 and is frozen during stage 2, and vice versa.

Y. Tian, H. Song, Z. Wang, H. Wang, Z. Hu, F. Wang, N. V. Chawla, and P. Xu, “Graph neural prompting with large language models,” 2023. [Online]. Available: <https://arxiv.org/abs/2309.15427>

Title: Graph Neural Prompting with LLMs

This study is conducted in the context of knowledge graphs and focuses on question answering tasks. The process begins by tokenizing the concatenation of the question, optional context, and answer options into a sequence of tokens, denoted as X. Relevant subgraphs are then retrieved by identifying entities in the knowledge graph that are related to tokens in X, including their 2-hop neighbors. To generate knowledge graph representations, a GNN-based encoder is employed: node embeddings are first initialized using pre-trained embeddings, and then refined using a Graph Attention Network (GAT). Finally, self-supervised link prediction is applied to extract relational information between entities, enabling the model to better understand and reason over the structured knowledge.



7.

D. Wang, Y. Zuo, F. Li, and J. Wu, “Llms as zero-shot graph learners: Alignment of gnn representations with llm token embeddings,” 2024. [Online]. Available: <https://arxiv.org/abs/2408.14512>

Title: *LLM as Zero-Shot Graph Learners: Alignment of GNN Representation with LLM Token Embedding*

This study aims to enable zero-shot graph learning using large language models (LLMs) by aligning the output of graph neural networks (GNNs) with LLM token embeddings, all without requiring additional training or fine-tuning of the LLM. The approach supports both node classification and link prediction tasks while ensuring that GNN outputs are usable as language tokens, eliminating the need to fine-tune the LLM for each task or dataset. The workflow consists of four main stages: (1) Pretraining a GNN using dual contrastive learning, which includes instance-wise contrastive learning—where two augmented graph views (via edge removal and feature masking) are generated to ensure structural consistency—and feature-wise contrastive learning, which aligns GNN features column-wise with LLM token embeddings via PCA to bridge the semantic space of the GNN and the text space of the LLM. (2) Training a simple linear projector to map each node or edge embedding into a fixed number of graph tokens that match the dimensionality of LLM tokens. (3) Designing unified natural language instructions to construct prompt templates that include graph token embeddings, task descriptions, input node or edge titles, and answer options. (4) Using the LLM for zero-shot predictions, keeping the model entirely frozen. In this framework, the GNN becomes not only structure-aware—capturing graph topology—but also semantically aligned with the way LLMs represent information. Specifically, in instance-wise contrastive learning, two graph augmentations are created by randomly removing edges or masking node features, and contrastive loss is used to encourage consistency. In feature-wise contrastive learning, rather than comparing graph instances, feature dimensions (columns of embedding matrices) are aligned with the principal components of LLM token embeddings by projecting GNN features into that space.

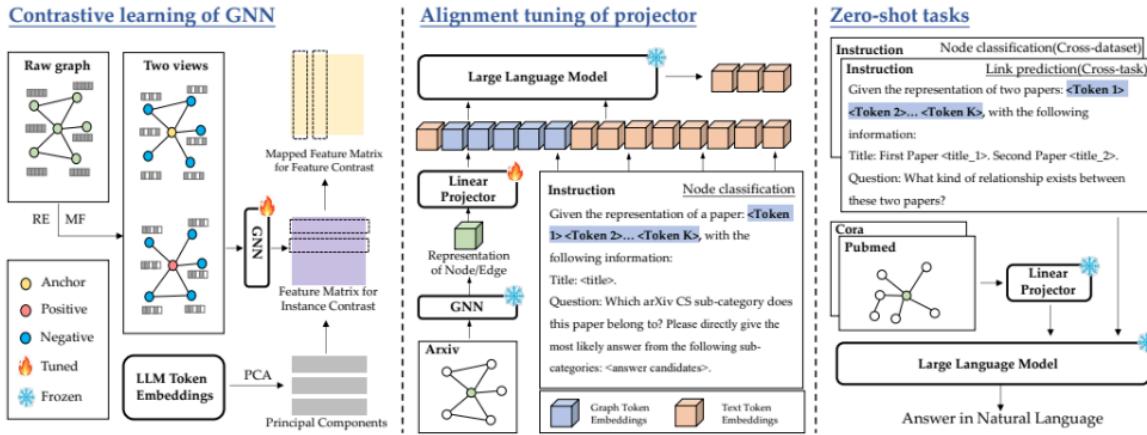


Figure 1: Framework of TEA-GLM

F. Hamann, M. Falk, and L. Walker, “Expanding knowledge graphs through text: Leveraging large language models for inductive link prediction,” in *INFORMATIK 2024*. Bonn: Gesellschaft für Informatik e.V., 2024, pp. 1407–1417.

Title: *Expanding KG through Text: Leveraging LLMs for Inductive Link Prediction*

In this study, a prompt is constructed by combining a generic system prompt with a task-specific template, optionally supplemented with related text or example outputs. The fully assembled prompt—whether composed of generic, specific, or supplemental content—is then passed to the large language model (LLM), which generates an output sequence. This output is subsequently parsed using a straightforward procedure that incrementally attempts to decode JSON objects containing mention strings. If any mentioned strings are successfully extracted, they are used to query and retrieve relevant vertices from the knowledge graph (KG), thereby grounding the language model’s predictions in the graph structure.

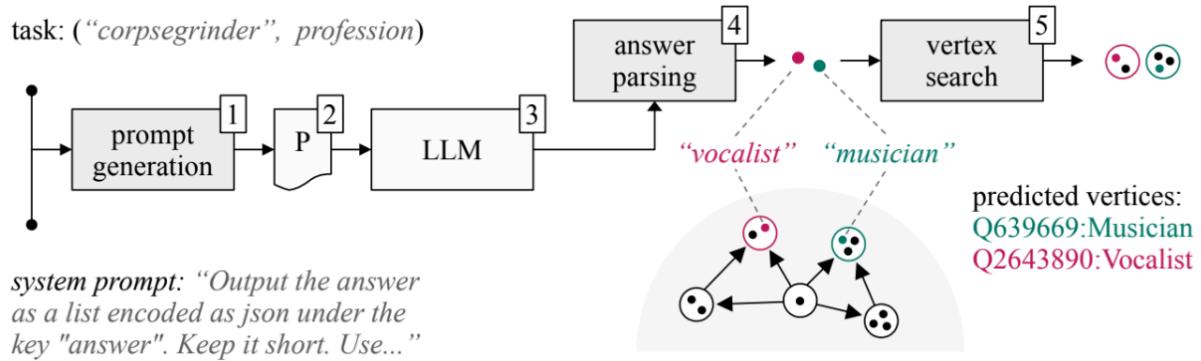


Fig. 2: Illustration of the full system. A task is sampled from the benchmark, verbalized and combined with a model specific system prompt. This prompt is fed to an LLM which is to answer with mention strings. These are used for a lookup in the known graph. All vertices which are associated with matched mentions are then returned as the model’s prediction.

9.

G. Park, P. Baity, B.-J. Yoon, and A. Hoisie, “Enhancing future link prediction in quantum computing semantic networks through llm-initiated node features,” 2024. [Online]. Available: <https://arxiv.org/abs/2410.04251>

Title: *Enhancing Future Link Prediction in Quantum Computing Semantic Networks*

In this study, initial node features are generated using large language models (LLMs) by prompting with queries such as “What is the feature of ...”. The resulting textual responses are then embedded into vector representations, which are subsequently used as input to a graph neural network (GNN) or a multi-layer perceptron (MLP) for downstream tasks.

Query: What are the features of quasiparticle in quantum computing?

Answer:

1. **Wave-Particle Duality:** Quasiparticles exhibit both wave-like and particle-like behavior. They can be described by a wave function, which determines their quantum state and allows them to interfere with each other. At the same time, they can also behave as localized particles with well-defined energy and momentum.
2. **Collective Behavior:** Quasiparticles often arise from the collective behavior of many interacting particles. They emerge as excitations of a quantum system, such as a solid or a liquid, and represent the quantum mechanical description of these collective modes.
3. **Quasiparticle Energy and Momentum:** Quasiparticles have a well-defined energy and momentum, which are related to their wave-like properties. The energy and momentum of a quasiparticle are typically not the same as the energy and momentum of the individual particles that make up the system.

...

Y. Li, Z. Li, P. Wang, J. Li, X. Sun, H. Cheng, and J. X. Yu, “A survey of graph meets large language model: Progress and future directions,” 2024. [Online]. Available: <https://arxiv.org/abs/2311.12399>

Title: *A Survey of Graph Meets Large Language Model: Progress and Future Directions*
LLM applications in graph learning are categorized into:

- **Enhancement Approaches:** Using LLMs to enrich node embeddings via explanations or text-derived embeddings.
- **Prediction Approaches:**
 - *Flatten-based*: Convert graph to sequences for textual processing.
 - *GNN-based*: Combine LLMs with structural understanding via GNNs.
- **GNN-LLM Alignment:**
 - *Symmetric*: Balanced alignment of modalities.
 - *Asymmetric*: GNNs support LLMs’ structure comprehension.

Future Directions Identified:

1. Extending beyond TAGs,
2. Addressing data leakage,
3. Enhancing transferability and explainability,
4. Improving efficiency and expressiveness.

11.

Z. Chen, H. Mao, H. Li, W. Jin, H. Wen, X. Wei, S. Wang, D. Yin, W. Fan, H. Liu, and J. Tang, “Exploring the potential of large language models (llms) in learning on graphs,” 2024. [Online]. Available: <https://arxiv.org/abs/2307.03393>

Title: *Exploring the potential of large language models (LLMs) in learning on Graphs*

The paper explores two primary methodologies for integrating Large Language Models (LLMs) into graph learning, specifically for node classification in text-attributed graphs (TAGs). The first pipeline, LLMs-as-Enhancers, uses embedding-visible LLMs (e.g., BERT, Sentence-BERT) to enhance node features either at the feature level (by generating embeddings) or at the text level (via augmentation using pseudo-labels or knowledge, such as TAPE and KEA). These enhanced features are then fed into Graph Neural Networks (GNNs). Within this, an iterative co-training strategy (GLEM) is also proposed, where GNNs and LLMs improve each other through pseudo-labeling. The second pipeline, LLMs-as-Predictors, directly employs

embedding-invisible LLMs like ChatGPT to make predictions by prompting them with natural language descriptions of both node attributes and graph structures. The study evaluates these pipelines across various datasets and settings to assess their effectiveness and scalability.

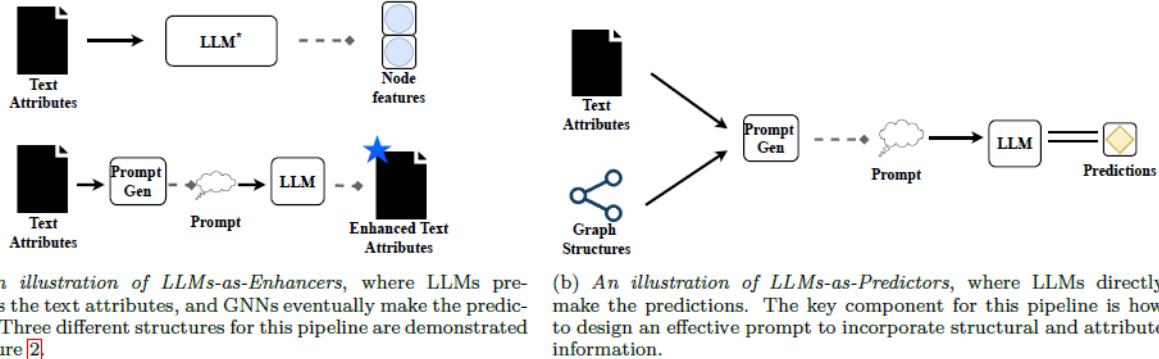


Figure 1: Pipelines for integrating LLMs into graph learning. In all figures, we use “PLM” to denote small-scale PLMs that can be fine-tuned on downstream datasets, “LLM” to denote embedding-visible LLMs, and “LLM” to denote embedding-invisible LLMs.

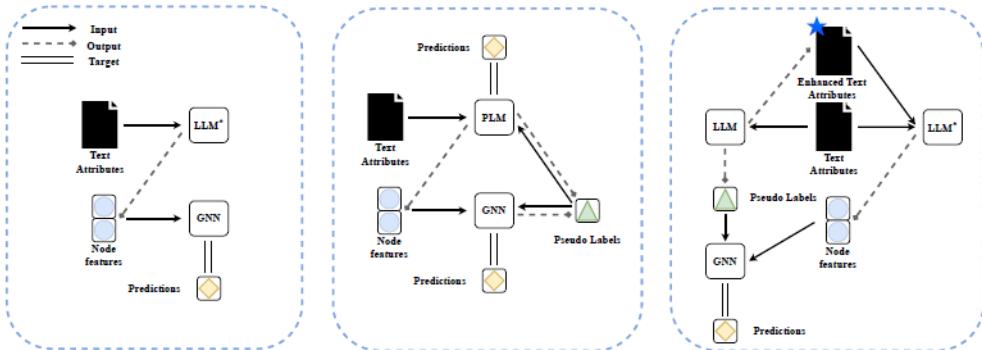


Figure 2: Three strategies to adopt LLMs as enhancers. The first two integrating structures are designed for feature-level enhancement, while the last structure is designed for text-level enhancement. From left to right: (1) Cascading Structure: Embedding-visible LLMs enhance text attributes directly by encoding them into initial node features for GNNs. (2) Iterative Structure: GNNs and PLMs are co-trained in an iterative manner. (3) Text-level enhancement structure: Embedding-invisible LLMs are initially adopted to enhance the text attributes by generating augmented attributes. The augmented attributes and original attributes are encoded and then ensembled together.

12.

C. Qian, H. Tang, Z. Yang, H. Liang, and Y. Liu, “Can large language models empower molecular property prediction?” 2023. [Online]. Available: <https://arxiv.org/abs/2307.07443>

Title: *Can Large Language Models Empower Molecular Property Prediction?*

The paper introduces a novel methodology that leverages Large Language Models (LLMs), particularly ChatGPT, for molecular property prediction through two main strategies: zero/few-shot classification and a new representation technique called Captions as new Representation (CaR). First, they use in-context learning by prompting ChatGPT with SMILES strings and corresponding class labels to perform classification without model fine-tuning. Second, they generate rich textual descriptions of molecules—capturing features like functional groups and chemical properties—using ChatGPT, and then fine-tune small-scale language models (e.g., RoBERTa, DeBERTa) on these captions for downstream molecular prediction tasks. This approach transforms molecular data into semantically meaningful text representations, enabling competitive performance compared to traditional GNN- and SMILES-based models.

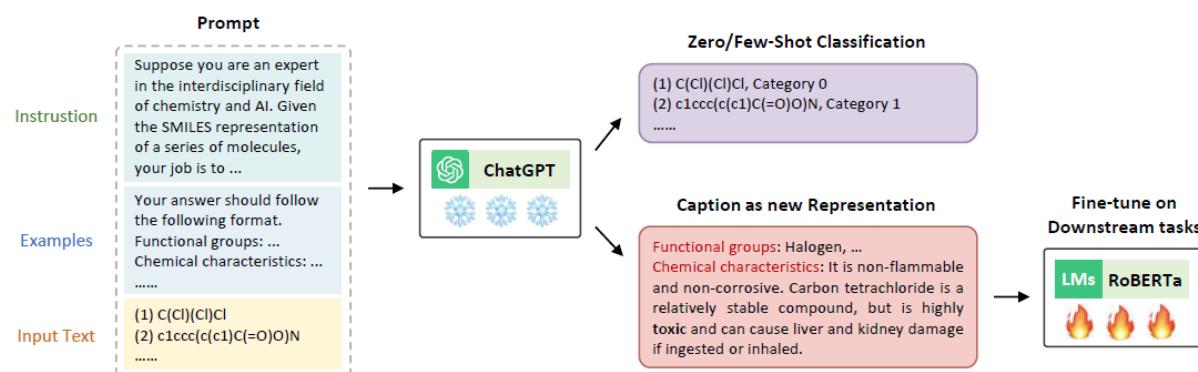


Figure 2: Overview of LLM4Mol.

13.

W. Wei, X. Ren, J. Tang, Q. Wang, L. Su, S. Cheng, J. Wang, D. Yin, and C. Huang, “Llmrec: Large language models with graph augmentation for recommendation,” 2024. [Online]. Available: <https://arxiv.org/abs/2311.00423>

Title: *LLMRec: Large Language Models with Graph Augmentation for Recommendation*

The LLMRec framework enhances recommendation systems by using Large Language Models (LLMs) to augment the user-item interaction graph through three core strategies: (1) generating additional user-item interaction edges via an LLM-based Bayesian Personalized Ranking sampler, (2) enriching item attributes and user profiles using prompts that infer missing or noisy side information, and (3) integrating the augmented data into the recommendation graph through feature projection, GNN-based context injection, and embedding fusion. To ensure robustness, it

introduces a denoised data mechanism that prunes unreliable interactions and applies Masked Autoencoders to refine noisy features. This entire process creates a richer, semantically informed graph structure that improves recommendation accuracy in sparse or low-quality data scenarios.

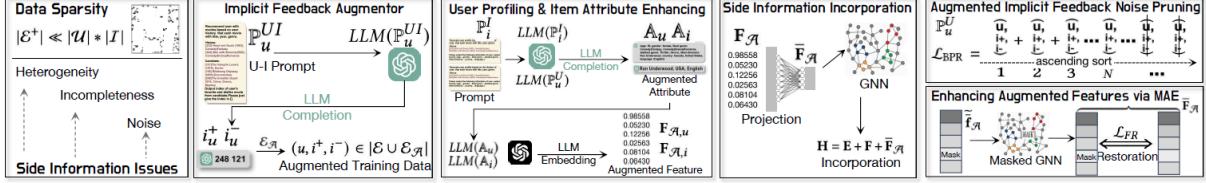


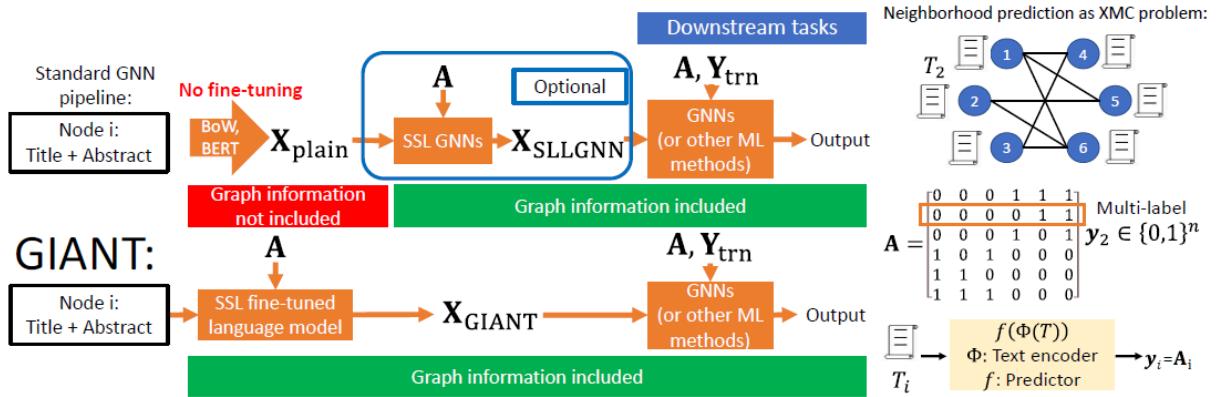
Figure 1: The LLMRec framework: (1) Three types of data augmentation strategies: i) augmenting user-item interactions; ii) enhancing item attributes, and iii) user profiling. (2) Augmented training with and denoised data robustification mechanism.

14.

E. Chien, W.-C. Chang, C.-J. Hsieh, H.-F. Yu, J. Zhang, O. Milenkovic, and I. S. Dhillon, “Node feature extraction by self-supervised multi-scale neighborhood prediction,” 2022. [Online]. Available: <https://arxiv.org/abs/2111.00064>

Title: *Node Feature Extraction by Self-Supervised Multi-Scale Neighborhood Prediction*

The paper proposes GIANT, a self-supervised learning framework for extracting graph-aware node features directly from raw data (like text), addressing the limitations of graph-agnostic feature extraction in standard GNN pipelines. The core methodology involves a novel neighborhood prediction task, where each node's raw text is used to predict its 1-hop neighbors, formulated as an Extreme Multi-label Classification (XMC) problem. To solve this efficiently at scale, GIANT employs XR-Transformers, which use a coarse-to-fine label clustering tree to guide the fine-tuning of a Transformer-based encoder (e.g., BERT). This encoder then produces node embeddings that capture both semantic and structural information, which are used in downstream tasks with or without GNNs.

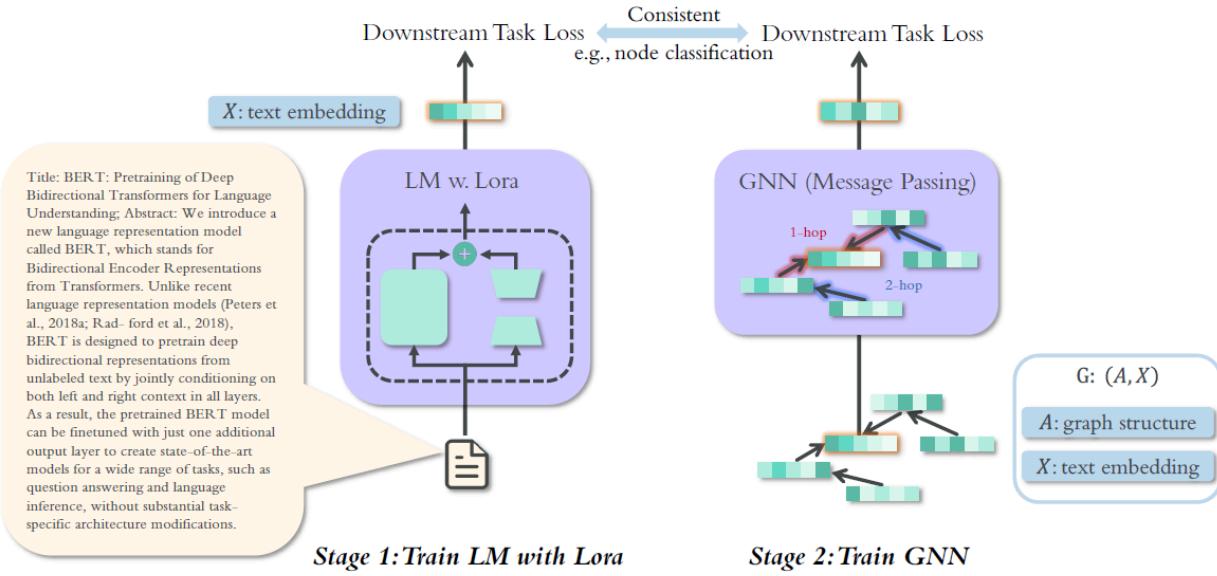


15.

K. Duan, Q. Liu, T.-S. Chua, S. Yan, W. T. Ooi, Q. Xie, and J. He, “Simteg: A frustratingly simple approach improves textual graph learning,” 2023. [Online]. Available: <https://arxiv.org/abs/2308.02565>

Title: SimTeG: A Frustratingly Simple Approach Improves Textual Graph Learning

The methodology of SimTeG follows a simple two-stage workflow designed to improve textual graph learning without architectural complexity. In the first stage, a pre-trained language model (LM) is fine-tuned using parameter-efficient techniques (e.g., LoRA) on the downstream task (such as node classification or link prediction), and node embeddings are generated from the LM's last hidden states using mean pooling. In the second stage, these fixed embeddings are combined with the graph structure and used as input to a Graph Neural Network (GNN), which is trained independently on the same task. This decoupled approach leverages the representational power of fine-tuned LMs while retaining the structural advantages of GNNs, enabling strong performance with simple models.

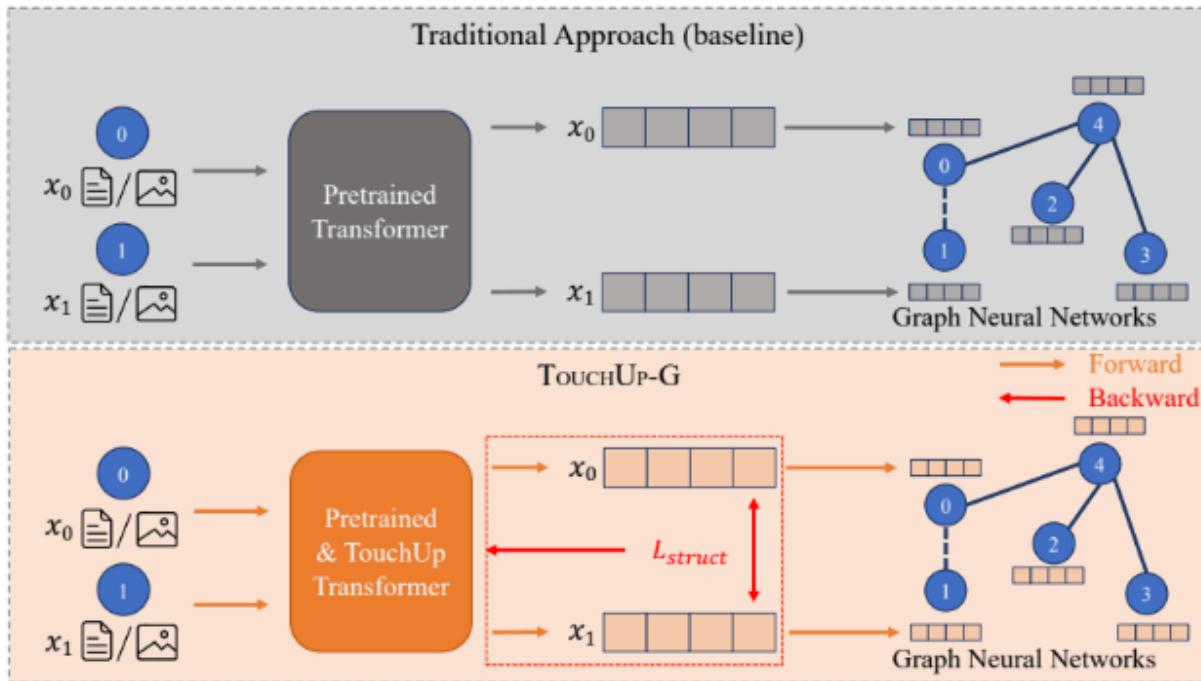


16.

J. Zhu, X. Song, V. N. Ioannidis, D. Koutra, and C. Faloutsos, “Touchup-g: Improving feature representation through graph-centric finetuning,” 2025. [Online]. Available: <https://arxiv.org/abs/2309.13885>

Title: *TouchUp-G: Improving Feature Representation through Graph-Centric Finetuning*

The paper proposes TouchUp-G, a simple and general "Detect & Correct" framework that enhances node features from pretrained models (PMs) like BERT or ViT for graph learning tasks. First, it introduces a feature homophily score to detect misalignment between the graph structure and node features extracted from PMs. If the score indicates poor alignment, TouchUp-G applies a graph-centric finetuning step that adjusts the PM by minimizing a structural loss, encouraging connected nodes to have more similar features. This corrected feature representation is then used as input to Graph Neural Networks (GNNs) for downstream tasks such as node classification or link prediction, resulting in significantly improved performance across various datasets and modalities.



17.

Y. Tan, Z. Zhou, H. Lv, W. Liu, and C. Yang, “Walklm: A uniform language model fine-tuning framework for attributed graph embedding,” in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 13 308–13 325.

Title: *WalkLM: A Uniform Language Model Fine-tuning Framework for Attributed Graph Embedding*

The methodology of the WalkLM framework centers on transforming attributed graphs into a textual format suitable for language models by combining attributed random walks with a rule-based textualization process. Specifically, it first performs random walks on the graph that capture both structural and attribute information, then automatically converts the resulting sequences into natural language-like text by embedding node and edge attributes into readable templates. These sequences form a corpus used to fine-tune a pre-trained language model (e.g., DistilRoBERTa) through masked language modeling, enabling the model to learn rich semantic and structural representations of the graph. The resulting embeddings can then be directly applied to downstream tasks like node classification and link prediction in a fully unsupervised and generalizable manner.

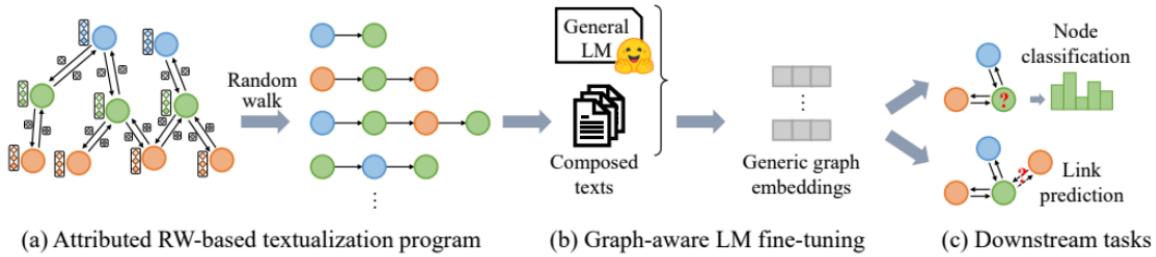


Figure 2: **The overall framework of WalkLM:** (a) An attributed random walk (RW) based textualization program obtains attributed RWs and composes roughly meaningful textual sequences directly from the attributed RWs. (b) A pre-trained general LM is fine-tuned in a graph-aware fashion using the generated textual sequences to produce generic graph embeddings. (c) The learned graph embeddings can be applied to various downstream tasks.

18.

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,” 2024. [Online]. Available: <https://arxiv.org/abs/2310.00149>

Title: *One for All: Towards Training One Graph Model for All Classification Tasks*

The authors propose OFA (One-for-All), a unified framework that enables a single graph neural network to perform classification tasks across diverse graph domains and task types. Their methodology centers around converting all graph data into Text-Attributed Graphs (TAGs), where nodes and edges are described using natural language, and encoded into a shared embedding space via a Large Language Model (LLM). They introduce the Nodes-of-Interest (NOI) concept to unify node-, link-, and graph-level tasks, and add a NOI prompt node with task-specific text. To support in-context learning, they develop a Graph Prompting Paradigm (GPP) that injects task descriptions and class information into the graph structure, enabling the model to adapt dynamically to various tasks—including supervised, few-shot, and zero-shot learning—without retraining.

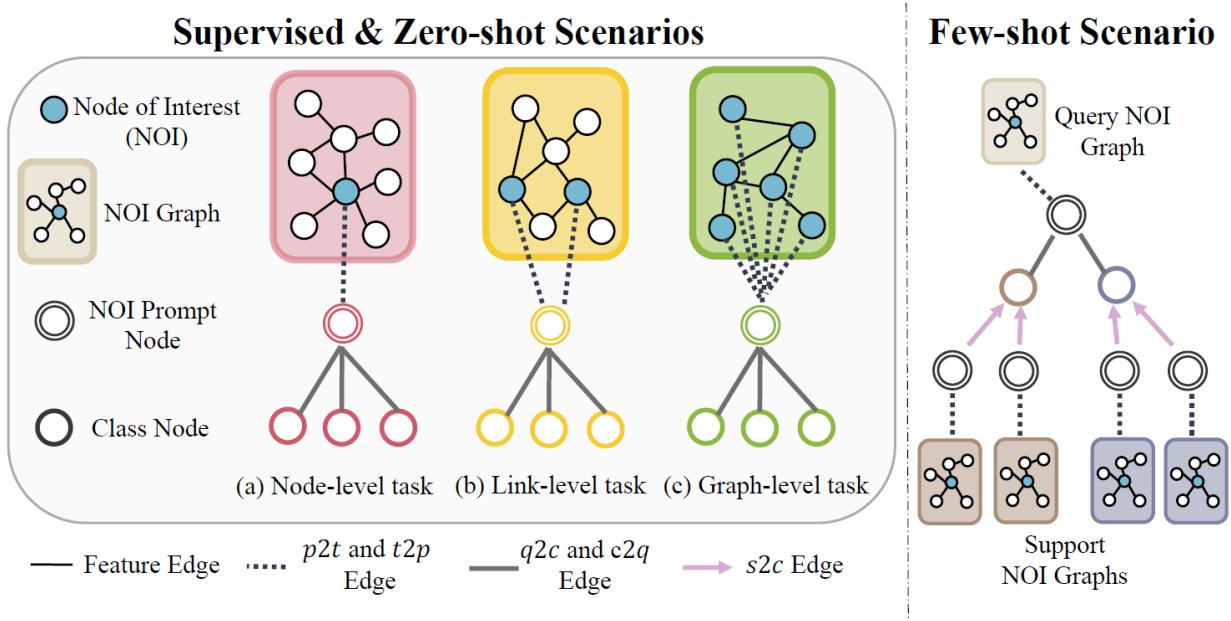
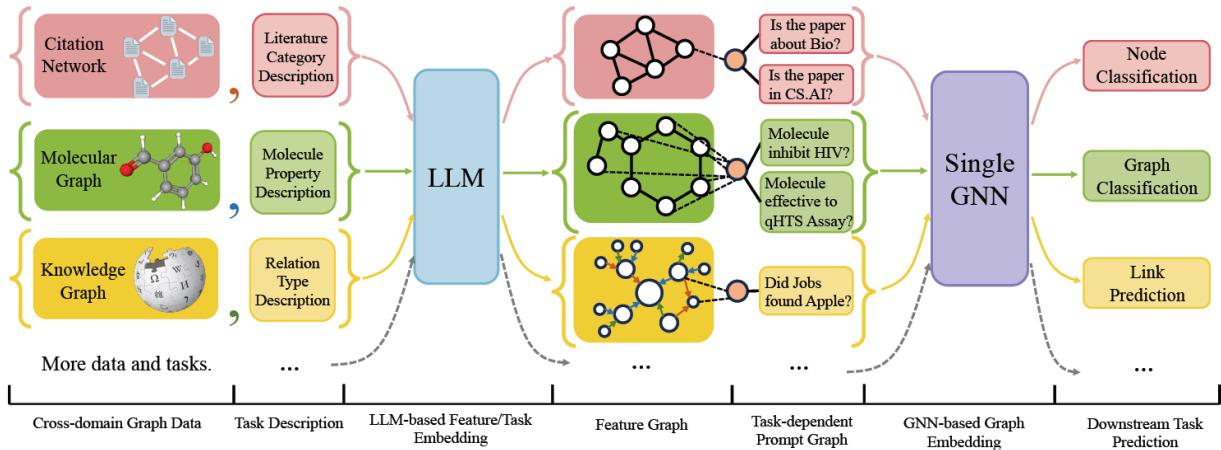


Figure 2: In-context learning design in OFA

19.

H. Xie, D. Zheng, J. Ma, H. Zhang, V. N. Ioannidis, X. Song, Q. Ping, S. Wang, C. Yang, Y. Xu, B. Zeng, and T. Chilimbi, “Graph-aware language model pre-training on a large graph corpus can help multiple graph applications,” 2023. [Online]. Available: <https://arxiv.org/abs/2306.02592>

Title: *Graph-Aware Language Model Pre-Training on a Large Graph Corpus Can Help Multiple Graph Applications*

The methodology of the paper revolves around GaLM, a graph-aware language model pre-training framework designed to learn from a large heterogeneous graph corpus rich in text and relational data. The workflow begins by pre-training large language models (e.g., BERT) on this graph using a link prediction task, allowing the LMs to incorporate structural knowledge. Two variants are explored: GaLM (LM-only training with graph supervision) and GaLM^{co} (co-training LMs with a GNN). For downstream tasks—such as link prediction or node classification on smaller application graphs with possibly different schemas—they fine-tune the pre-trained models using several strategies: direct LM fine-tuning, adding a task-specific GNN, and stitching the application graph with the large corpus to enrich relational context. This enables effective transfer of both textual and graph-based knowledge across varied graph applications.

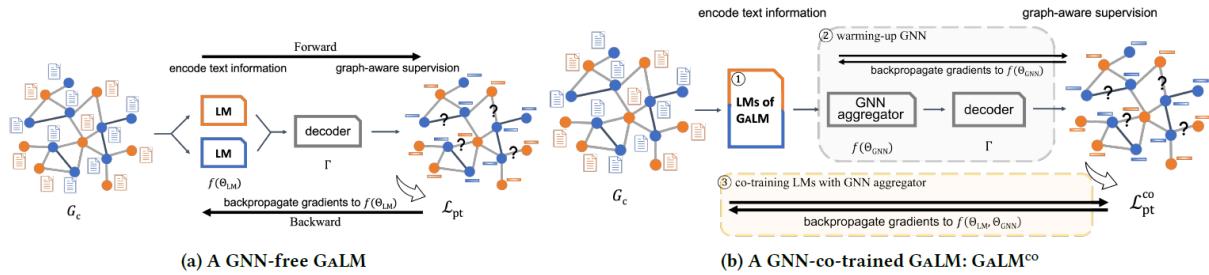


Figure 3: Graph-aware LM pre-training framework (GaLM). The LMs are pre-trained on a given large graph corpus either with or without the incorporation of GNN aggregators (GaLM^{co} or GaLM, respectively). (a) The pre-training of GaLM further trains existing general LMs in a graph-aware manner. (b) The pre-training of GaLM^{co} includes three steps: i. fine-tuning existing general LMs by graph-aware supervision, ii. warming up the GNN aggregator by fixing the graph-aware pre-trained LMs, iii. co-training the graph-aware pre-trained LMs with the warmed-up GNN aggregator by end-to-end backpropagating the loss to LMs through GNN.

20.

X. Huang, K. Han, D. Bao, Q. Tao, Z. Zhang, Y. Yang, and Q. Zhu, “Prompt-based node feature extractor for few-shot learning on text-attributed graphs,” 2023. [Online]. Available: <https://arxiv.org/abs/2309.02848>

Title: *Prompt-based Node Feature Extractor for Few-shot Learning on Text-Attributed Graphs*

The paper proposes G-Prompt, a two-stage framework for extracting task-specific and graph-aware node features from Text-Attributed Graphs (TAGs). In the first stage, it trains a Graph Adapter—a lightweight GNN layer placed on top of a pre-trained language model (PLM)—using a self-supervised fill-mask task, allowing the PLM to incorporate graph structure by adjusting predictions based on neighbors’ context. In the second stage, task-specific prompts are prepended to the node texts, and the adapter-enhanced PLM predicts the masked tokens.

These predictions are then transformed into node feature vectors, capturing both the graph's structural information and the downstream task semantics, which can be used for few-shot or zero-shot learning.

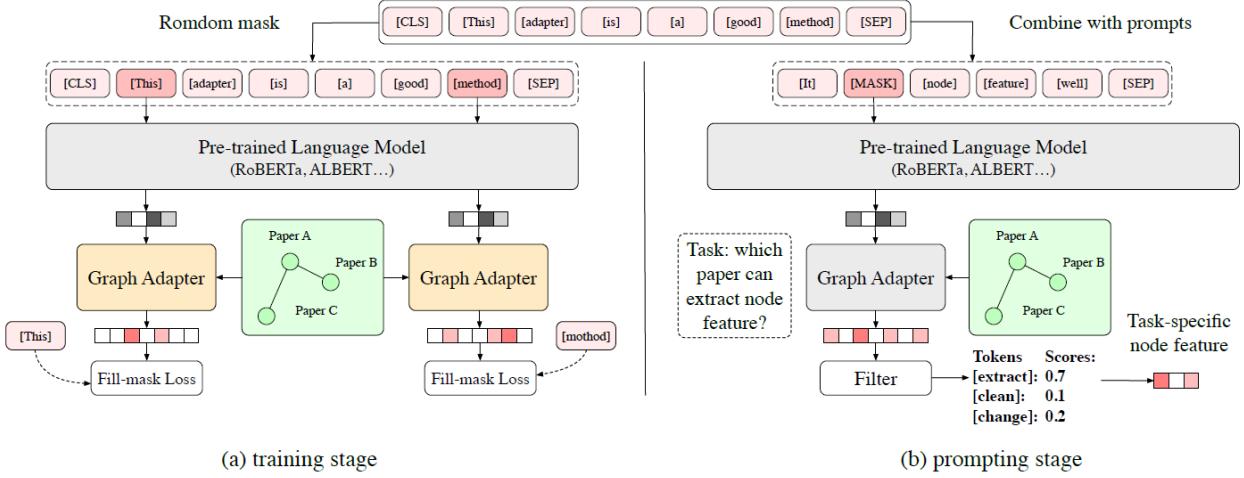


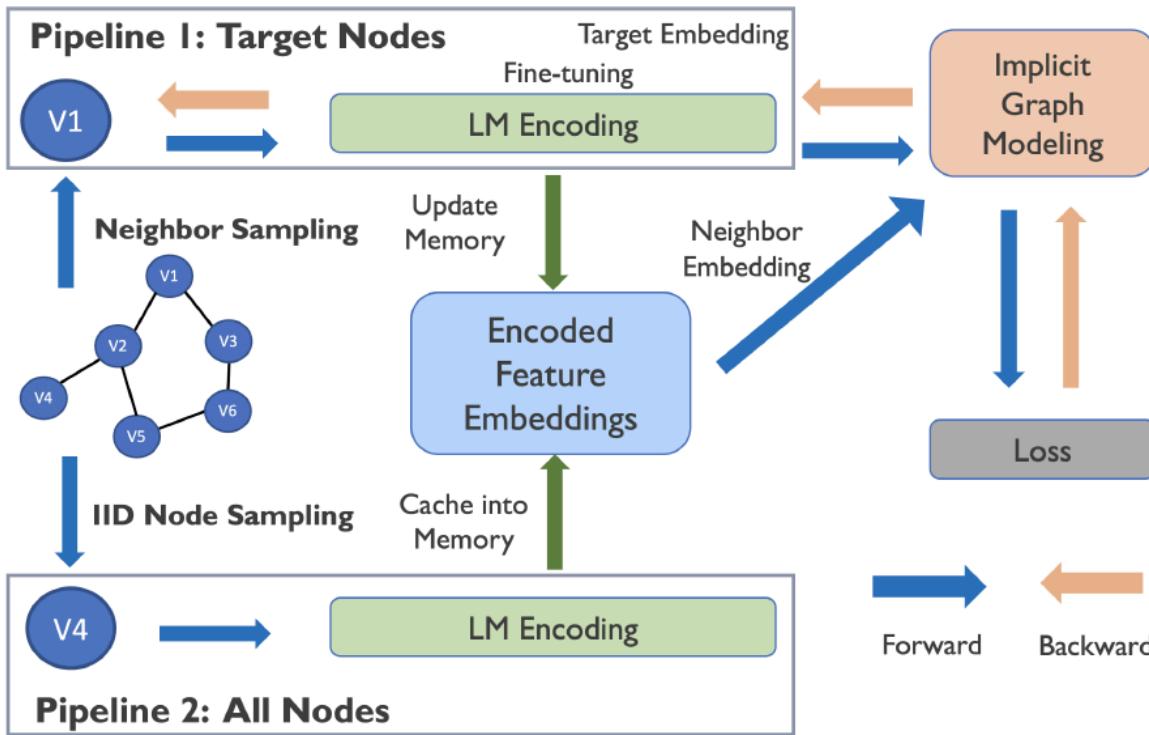
Figure 1: Framework of G-Prompt

21.

R. Xue, X. Shen, R. Yu, and X. Liu, “Efficient end-to-end language model fine-tuning on graphs,” 2024. [Online]. Available: <https://arxiv.org/abs/2312.04737>

Title: *Efficient End-to-end Language Model Fine-tuning on Graphs*

The paper proposes LEADING, a novel and efficient end-to-end fine-tuning framework that integrates Language Models (LMs) with Graph Neural Networks (GNNs) for learning on text-attributed graphs. To overcome the challenges of computational and data inefficiency in existing methods, LEADING introduces a two-pipeline training workflow: one pipeline encodes only target nodes with gradient updates, while a second encodes neighbor nodes without gradients and caches their embeddings. This neighbor decoupling reduces redundant LM computations. Additionally, LEADING replaces deep GNN stacks with APPNP-based implicit graph modeling, which computes node embeddings through an iterative fixed-point propagation, enabling efficient long-range information capture without high memory costs. Together, these components allow for scalable and effective joint LM-GNN training with minimal labeled data.



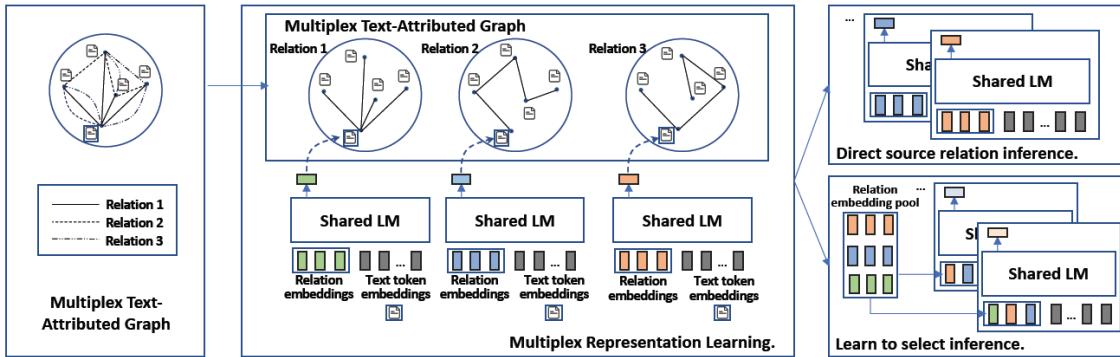
22.

B. Jin, W. Zhang, Y. Zhang, Y. Meng, H. Zhao, and J. Han, “Learning multiplex representations on text-attributed graphs with one language model encoder,” 2024. [Online]. Available: <https://arxiv.org/abs/2310.06684>

Title: *Learning Multiplex Representations on Text-Attributed Graphs with One Language Model Encoder*

The paper proposes METAG, a framework for learning relation-specific embeddings on multiplex text-attributed graphs using a single shared language model encoder. Instead of training separate models for each relation type, METAG introduces relation prior tokens—learnable tokens that are prepended to the input text to encode relation-specific semantics. The shared encoder learns general language knowledge, while these tokens enable the model to produce different embeddings per relation for the same node. During training, METAG uses a contrastive learning objective with negative sampling to ensure that related nodes have similar embeddings under the correct relation. For downstream tasks, METAG supports direct inference when the relevant relation is known and a learn-to-select mechanism using attention

over relation embeddings when it is not, allowing flexible and efficient use of the learned representations.

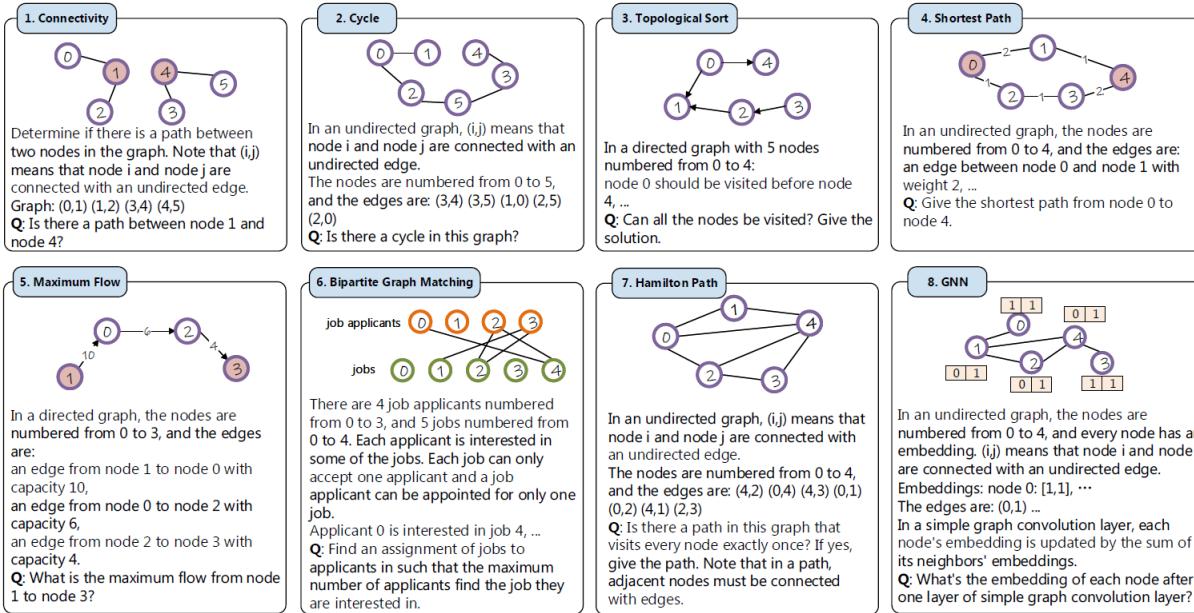


23.

H. Wang, S. Feng, T. He, Z. Tan, X. Han, and Y. Tsvetkov, “Can language models solve graph problems in natural language?” 2024. [Online]. Available: <https://arxiv.org/abs/2305.10037>

Title: *Can Language Models Solve Graph Problems in Natural Language?*

In this paper, the authors introduce NLGraph, a large benchmark of 29,370 graph reasoning problems described in natural language, covering tasks from simple (connectivity, cycle detection) to complex (maximum flow, Hamiltonian path, GNN simulation). They generate synthetic graphs of varying sizes and sparsity, craft natural language questions, and control task difficulty. To test whether language models can solve these graph problems, they evaluate GPT-3, GPT-3.5, and GPT-4 using different prompting strategies, including zero-shot, few-shot, chain-of-thought (CoT), and advanced methods like self-consistency (SC) and least-to-most (LTM). To enhance model performance, they propose two new prompting techniques—Build-a-Graph Prompting (encouraging LLMs to mentally construct graphs first) and Algorithmic Prompting (embedding algorithm explanations like DFS into prompts)—then measure improvements across tasks to assess reasoning ability.

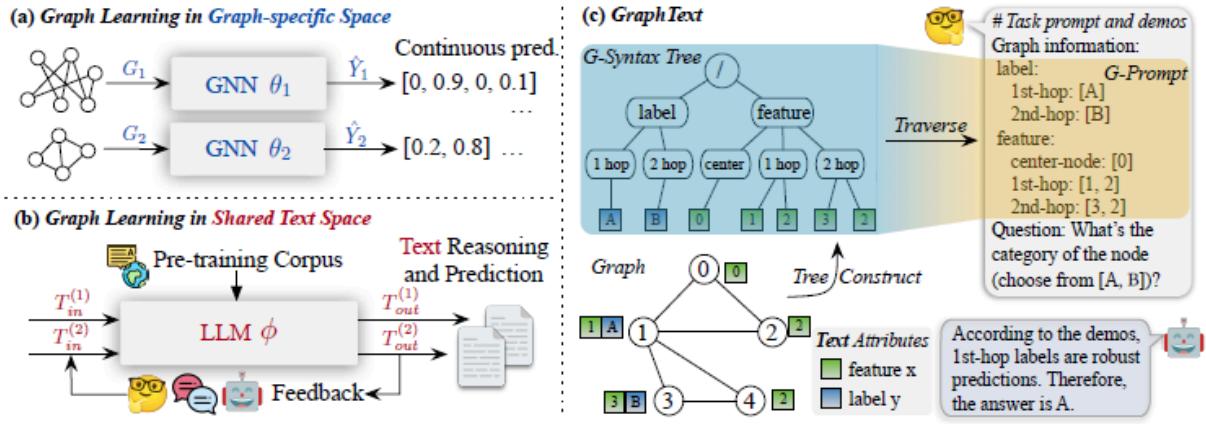


24.

J. Zhao, L. Zhuo, Y. Shen, M. Qu, K. Liu, M. Bronstein, Z. Zhu, and J. Tang, “Graphtext: Graph reasoning in text space,” 2023. [Online]. Available: <https://arxiv.org/abs/2310.01089>

Title: GRAPHTEXT: Graph Reasoning in Text Space

The authors propose GRAPHTEXT, a method that enables large language models (LLMs) to perform graph reasoning by converting graph data into natural language. Instead of using specialized graph neural networks (GNNs), GRAPHTEXT builds a graph-syntax tree that organizes both node attributes and relationships hierarchically. By traversing this tree, the graph structure is flattened into a text sequence ("graph prompt") that an LLM can read and reason about. Graph tasks like node classification are thus reformulated as text generation problems, allowing pretrained LLMs (e.g., ChatGPT) to perform training-free, in-context graph reasoning without fine-tuning. This method also supports interactive reasoning with humans and can generalize across different types of graphs.

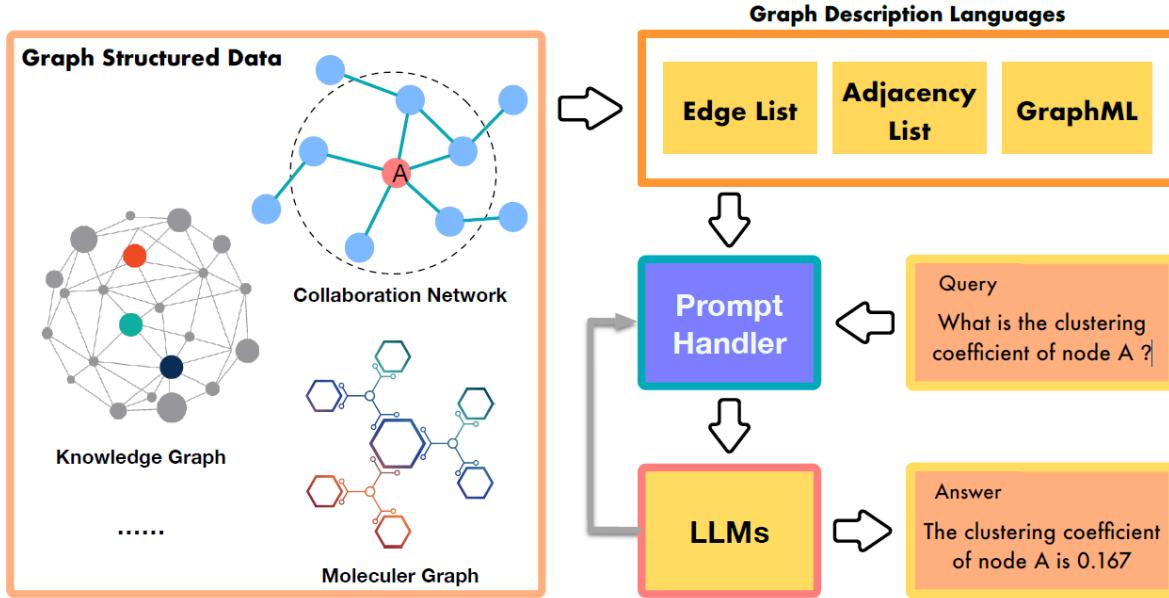


25.

J. Guo, L. Du, H. Liu, M. Zhou, X. He, and S. Han, “Gpt4graph: Can large language models understand graph structured data ? an empirical evaluation and benchmarking,” 2023. [Online]. Available: <https://arxiv.org/abs/2305.15066>

Title: GPT4Graph: Can Large Language Models Understand Graph Structured Data?

The authors propose a framework to evaluate large language models (LLMs) on graph-structured data by first converting graphs into human-readable textual formats known as Graph Description Languages (e.g., GraphML, edge lists). These textual graph representations are then combined with user queries using a prompt handler, which crafts input prompts for the LLMs. The methodology includes two prompting strategies: manual prompting, where format explanations, role instructions, and examples are carefully engineered to guide the model, and self-prompts, where the LLM generates intermediate context (like summaries or format clarifications) to improve its understanding before answering the main query. This workflow is applied across a range of structural and semantic graph tasks to empirically assess the LLMs’ reasoning capabilities.



26.

Y. Hu, Z. Zhang, and L. Zhao, “Beyond text: A deep dive into large language models’ ability on understanding graph data,” 2023. [Online]. Available: <https://arxiv.org/abs/2310.04944>

Title: *Beyond Text: A Deep Dive into Large Language Models’ Ability on Understanding Graph Data*

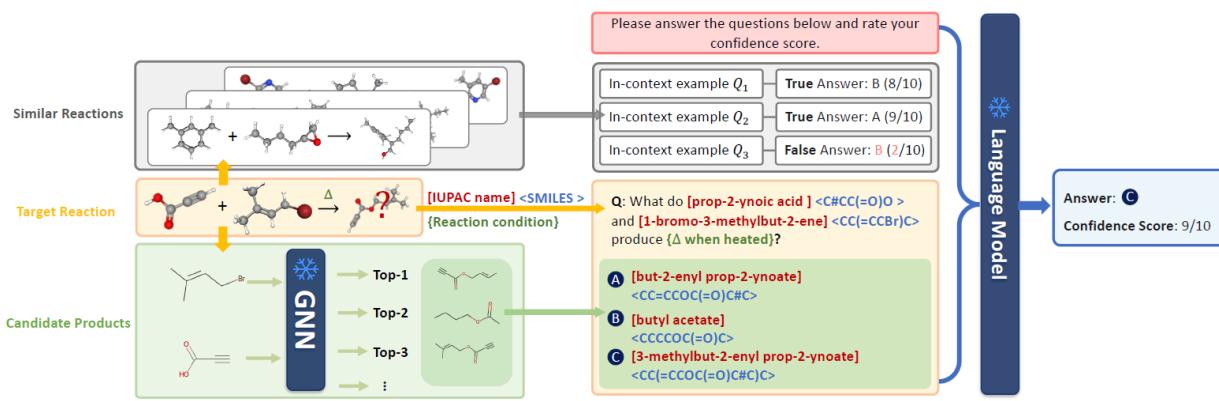
The authors evaluate the capability of large language models (LLMs), specifically GPT-3.5 and GPT-4, to perform graph-based learning tasks across node, edge, and graph levels by designing a systematic benchmarking workflow. They use five benchmark datasets (CORA, PUBMED, OGBN-ARXIV, WORDNET18, REDDIT) and formulate various prompt strategies—zero-shot, few-shot, with and without explicit structural (graph topology) information. For each task type, they craft tailored input prompts to test whether LLMs can interpret and utilize graph-related context to make accurate predictions. The models’ performances are then compared with traditional graph neural networks (GNNs) such as GCN and GAT, offering insights into how well LLMs can function in tasks typically reliant on structured graph representation.

27.

Y. Shi, A. Zhang, E. Zhang, Z. Liu, and X. Wang, “ReLM: Leveraging language models for enhanced chemical reaction prediction,” 2023. [Online]. Available: <https://arxiv.org/abs/2310.13590>

Title: *ReLM: Leveraging Language Models for Enhanced Chemical Reaction Prediction*

The ReLM framework integrates Graph Neural Networks (GNNs) and Language Models (LMs) to enhance chemical reaction prediction, especially in out-of-distribution cases. First, GNNs process the input reactants to generate a set of top-K candidate products based on molecular similarity. Then, a prompt is constructed for the LM that includes the reactants, reaction conditions, these candidate products, and a few in-context examples of similar reactions. The LM is tasked with selecting the most likely product and rating its confidence using a custom prompting technique called the Confidence Score Strategy (CSS), which improves interpretability and robustness by teaching the LM to self-assess based on both correct and intentionally incorrect examples. This method leverages the structural reasoning of GNNs with the textual understanding and inference capabilities of LMs.



28.

D. Das, I. Gupta, J. Srivastava, and D. Kang, “Which modality should i use – text, motif, or image? : Understanding graphs with large language models,” 2024. [Online]. Available: <https://arxiv.org/abs/2311.09862>

Title: *Which Modality Should I Use – Text, Motif, or Image? Understanding Graphs with Large Language Models*

The authors propose a methodology to evaluate how Large Language Models (LLMs), including GPT-4 and GPT-4V, understand graph-structured data by encoding graphs in three modalities:

text, motif, and image. For each modality, they transform graph structures into prompts suited for LLMs to perform node classification tasks. Text encodes nodes and edges as adjacency lists; motif encoding includes structural patterns like triangles and stars around a target node; image encoding visualizes the graph, leveraging GPT-4V's vision-language capabilities. They use three benchmark datasets (CORA, CITESEER, PUBMED), apply sampling techniques (ego graph and forest fire), and evaluate each modality using accuracy, mismatch rate, denial rate, and token usage. The authors also introduce the GRAPHTMI benchmark, which categorizes graph tasks by difficulty using homophily and motif count, to systematically analyze how each modality performs across varying structural complexities.

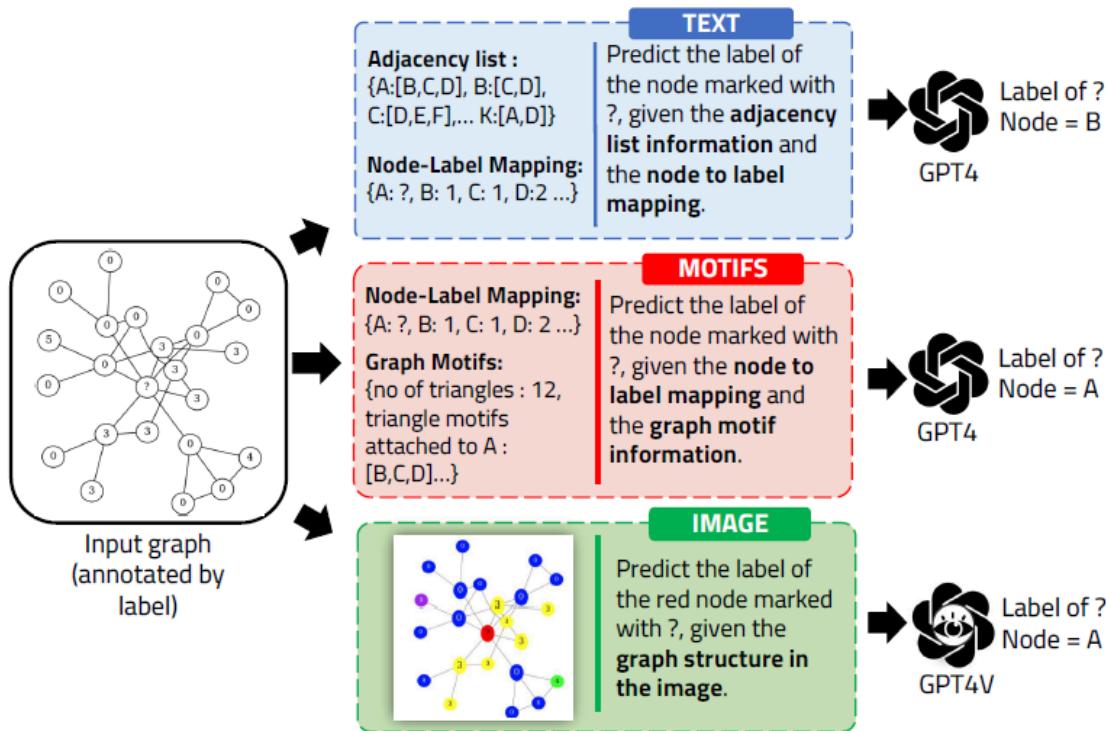


Figure 2: Node Classification on a Graph using different input modality encodings like Text, Motif, and Image.

29.

C. Liu and B. Wu, “Evaluating large language models on graphs: Performance insights and comparative analysis,” 2023. [Online]. Available: <https://arxiv.org/abs/2308.11224>

Title: *Evaluating Large Language Models on Graphs: Performance Insights and Comparative Analysis*

The authors evaluate large language models (LLMs) on their ability to understand and reason about graph-structured data by designing five core tasks—connectivity, node neighbor classification, node degree, pattern matching, and finding all shortest paths—using synthetic undirected graphs expressed as plain-text node and edge lists. They assess four LLMs (GPT-3.5-turbo, GPT-4, CalderaAI/30B-Lazarus, and Wizard-Vicuna-13B) under various prompting strategies: zero-shot, zero-shot chain-of-thought (CoT), and few-shot prompting. Each model's performance is evaluated across four key dimensions: comprehension, correctness, fidelity (reliability in multi-answer tasks), and rectification (ability to correct previous mistakes). This structured workflow enables a comprehensive comparison of LLM capabilities in processing graph data using natural language inputs.

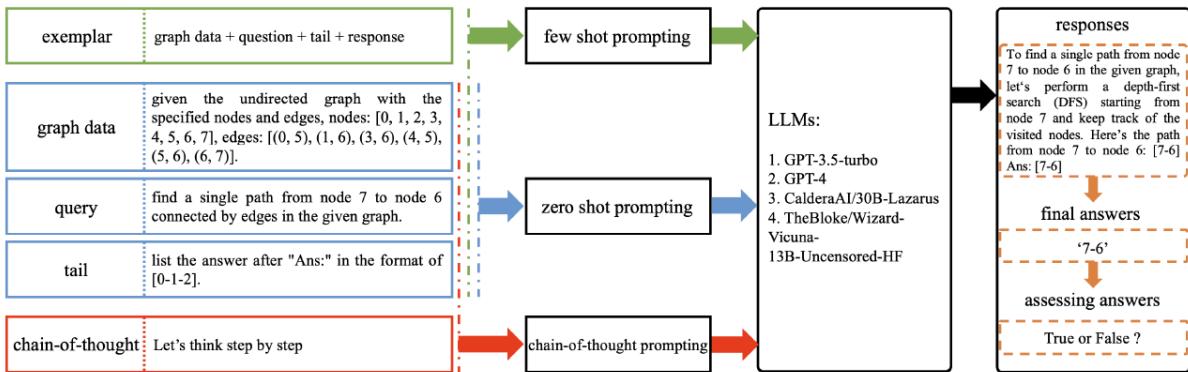


Figure 1: Example of Different Prompts.

30.

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?” 2024. [Online]. Available: <https://arxiv.org/abs/2310.17110>

Title: *LLM4DyG: Can Large Language Models Solve Spatial-Temporal Problems on Dynamic Graphs?*

The authors propose a comprehensive benchmark called LLM4DyG to evaluate the spatial-temporal reasoning abilities of large language models (LLMs) on dynamic graphs. Their methodology involves generating synthetic dynamic graphs using models like Erdős–Rényi, Stochastic Block, and Forest Fire, with controllable parameters such as node count, edge density, and time span. They design nine natural-language-based tasks that test different aspects of temporal, spatial, and combined reasoning on these graphs. Prompts are constructed using various strategies (zero-shot, one-shot, chain-of-thought), and the models’ responses are evaluated across different LLMs including GPT-3.5 and open-source models. To enhance

reasoning, they introduce Disentangled Spatial-Temporal Thoughts (DST2), a novel prompting technique that guides LLMs to process temporal and spatial information separately, leading to significant performance gains in complex tasks.

31.

J. Huang, X. Zhang, Q. Mei, and J. Ma, “Can llms effectively leverage graph structural information through prompts, and why?” 2024. [Online]. Available: <https://arxiv.org/abs/2309.16595>

Title: *Can LLMs Effectively Leverage Graph Structural Information through Prompts in Text-Attributed Graphs, and Why?*

The authors investigate how Large Language Models (LLMs) leverage structural information in text-rich graphs by designing a three-part methodology. First, to assess the impact of data leakage, they create a new benchmark dataset (arxiv-2023) containing papers published after the training cutoff of models like ChatGPT and LLaMA-2, and compare results with existing datasets. Second, to test whether LLMs interpret graph prompts structurally or as plain text, they conduct adversarial experiments by linearizing and rewiring the ego-graph structure in the prompts, observing negligible performance changes. Third, they analyze why structural information improves performance by examining homophily (label similarity among neighbors) and the richness of textual features; they find that LLMs benefit primarily from label-relevant phrases in homophilous neighborhoods, especially when the target node lacks sufficient text. Throughout, they use varied prompt styles to encode neighborhood information and test LLM responses in zero-shot and few-shot settings.

Prompt Style	Prompt Template
Zero-shot	Abstract: <abstract>\nTitle: <title>\nDo not give any reasoning or logic for your answer.\n\nAnswer: \n\n
Zero-shot CoT	Abstract: <abstract>\nTitle: <title>\nAnswer: \n\nLet's think step by step. \n
Few-shot	Abstract: <few-shot abstract>\n... \nAnswer: \n\n<few-shot label>\n... (more few-shot examples)\nAbstract: <abstract>... \nAnswer: \n\n
<i>k</i> -hop title, <i>k</i> -hop title+label	Abstract: <abstract>\nTitle: <title>\nIt has following neighbor papers at hop 1:\nPaper 1 title: <paper 1 title>\nLabel: <paper 1 label>\nIt is linked to paper <list of papers linked to paper 1>\n\nPaper 2 title: <paper 2 title>\nLabel: <paper 2 label>\nIt is linked to paper <list of papers linked to paper 2>\n\n... (more 1-hop neighbors)\n\nIt has following neighbor papers at hop 2:\n... (more 2-hop neighbors)\n\nDo not give any reasoning or logic for your answer. \n\nAnswer: \n\n
<i>k</i> -hop attention. Step 1: Attention extraction	The paper of interest is <title>. Please return a Python list of at most <k> indices of the most related papers among the following neighbors, ordered from most related to least related. If there are fewer than <k> neighbors, just rank the neighbors by relevance. The list should look like this: [1, 2, 3, ...]\n1: <neighbor title 1>\n... (more 1-hop neighbors) \n
<i>k</i> -hop attention. Step 2: Attention prediction	Abstract: <abstract>\nTitle: <title>\nIt has following important neighbors, from most related to least related:\n(more neighbors chosen by attention)\nDo not give any reasoning or logic for your answer. \n\nAnswer: \n\n
Linearized <i>k</i> -hop title, <i>k</i> -hop title+label:	Abstract: <abstract>\nTitle: <title>\n<paper 1 title>\nLabel: <paper 1 label>\n\n<paper 2 title>\nLabel: <paper 2 label>\n\n... (more 1-hop neighbors)\n... (more 2-hop neighbors)\n\nDo not give any reasoning or logic for your answer. \n\nAnswer: \n\n

32.

B. Fatemi, J. Halcrow, and B. Perozzi, “Talk like a graph: Encoding graphs for large language models,” 2023. [Online]. Available: <https://arxiv.org/abs/2310.04560>

Title: *Talk Like a Graph: Encoding Graphs for Large Language Models*

The paper proposes a methodology for enabling large language models (LLMs) to perform reasoning over graph-structured data by encoding graphs as natural language text without modifying the model architecture. The workflow involves two key components: a graph encoding function that transforms the graph structure into a textual format (e.g., adjacency lists, incident lists, or thematic relationship descriptions), and a prompt/question encoding function that frames graph-related queries in either technical or application-specific terms. These are combined into a single prompt and input to the LLM, treated as a black-box model. The study systematically evaluates various combinations of graph encodings and prompting strategies (zero-shot, few-shot, chain-of-thought, etc.) across multiple basic graph reasoning tasks using the proposed GraphQA benchmark, revealing how prompt structure, encoding choice, and graph topology influence LLM performance.

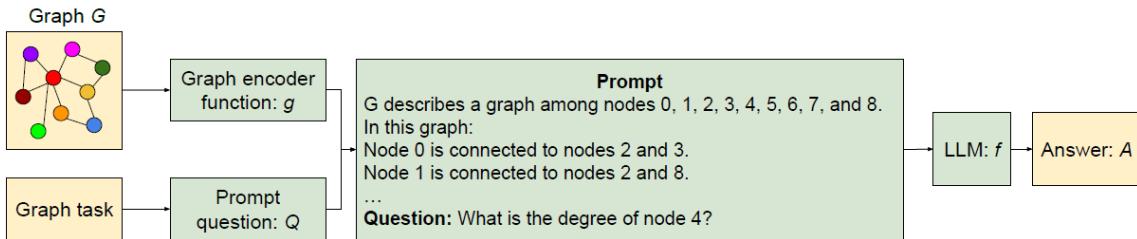


Figure 1: Overview of our framework for reasoning with graphs using LLMs.

33.

H. Zhao, S. Liu, C. Ma, H. Xu, J. Fu, Z.-H. Deng, L. Kong, and Q. Liu, “Gimlet: A unified graph-text model for instruction-based molecule zero-shot learning,” 2023. [Online]. Available: <https://arxiv.org/abs/2306.13089>

Title: *GIMLET: A Unified Graph-Text Model for Instruction-Based Molecule Zero-Shot Learning*

The authors propose GIMLET, a unified graph-text Transformer model designed for zero-shot molecule property prediction using natural language instructions. Instead of relying on labeled data or task-specific finetuning, GIMLET processes both molecular graphs and task instructions within a single language model by treating graph nodes and text tokens uniformly as input tokens. It introduces generalized position embeddings to encode both graph structure (via graph shortest distances) and text order, and uses a decoupled attention mechanism to separate graph encoding from instruction processing, enabling generalizable graph representations. During supervised pretraining, GIMLET learns from over 2,000 molecular tasks paired with descriptive instructions, allowing it to predict task outputs (classification or regression) directly from instructions and molecule data in a zero-shot or few-shot setting.

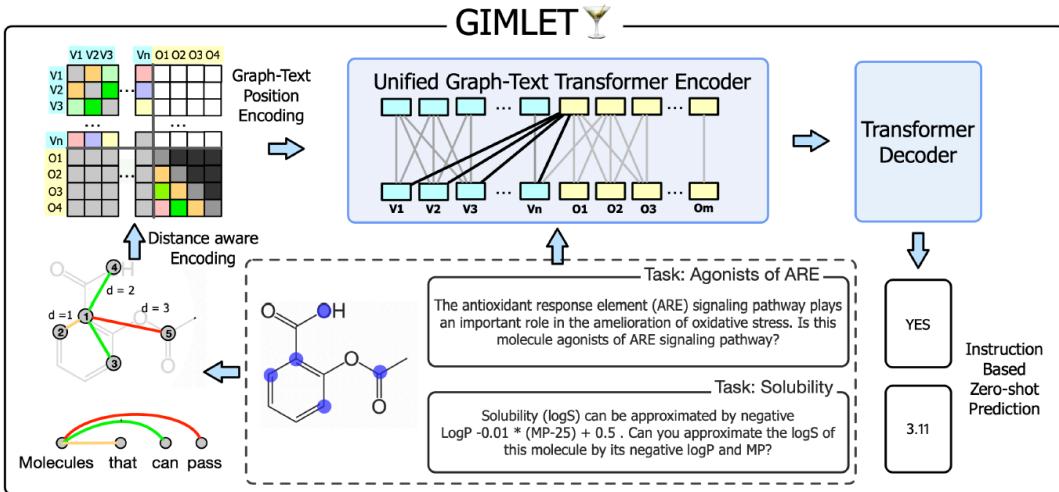


Figure 1: Our framework handles molecule tasks in the zero-shot fashion by natural language instruction. Within GIMLET, we employ distance-based joint position embedding to encode graphs and instruction texts. Additionally, we utilize attention masks to decouple the graph encoding process.

34.

R. Ye, C. Zhang, R. Wang, S. Xu, and Y. Zhang, “Language is all a graph needs,” 2024. [Online]. Available: <https://arxiv.org/abs/2308.07134>

Title: *Language is All a Graph Needs*

The authors propose a novel methodology that replaces traditional Graph Neural Networks (GNNs) with large language models (LLMs) by representing graph structures and tasks entirely through natural language. They design structured prompts that describe multi-hop neighborhoods, node/edge features, and paths using textual descriptions, and feed these into instruction-tuned LLMs (e.g., Flan-T5 or LLaMA). Each node is represented by a unique token with its embedding initialized from node features. The model is fine-tuned using a multi-task setup for node classification and a self-supervised link prediction task, with both tasks formulated as language generation problems. This approach unifies graph learning under a language modeling framework and eliminates the need for specialized graph architectures.

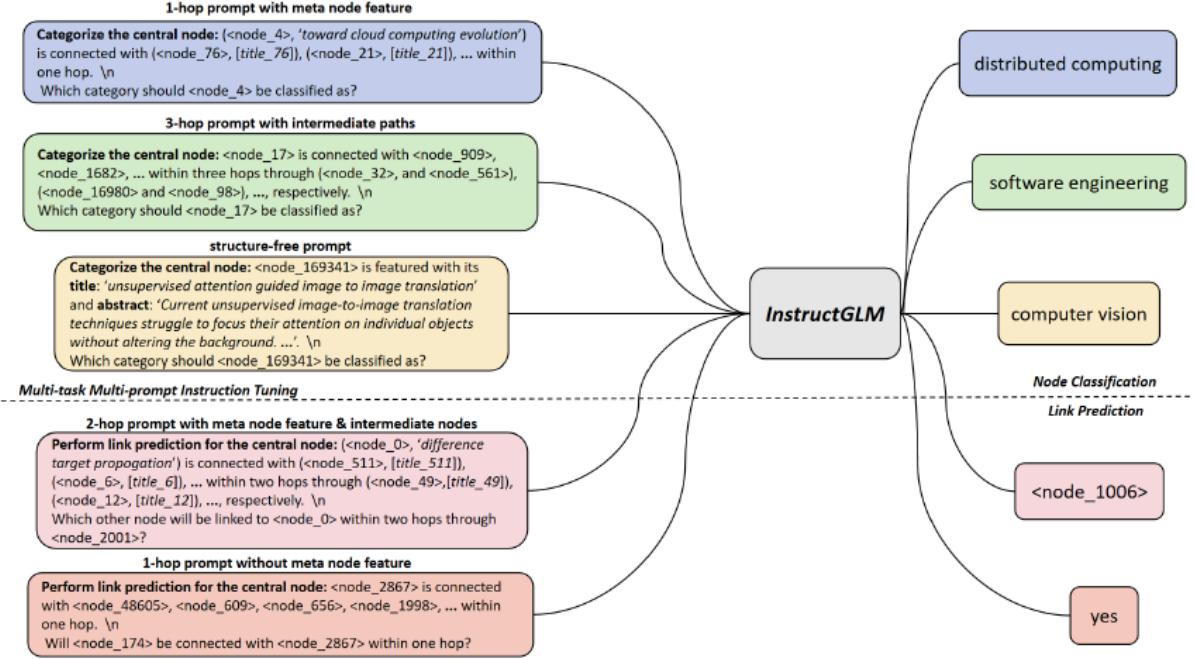


Figure 1: Illustration of the InstructGLM Framework. We fine-tune InstructGLM under a Multi-task Multi-prompt instruction tuning framework, enabling it to solve various graph machine learning tasks with the structure information purely described by natural language.

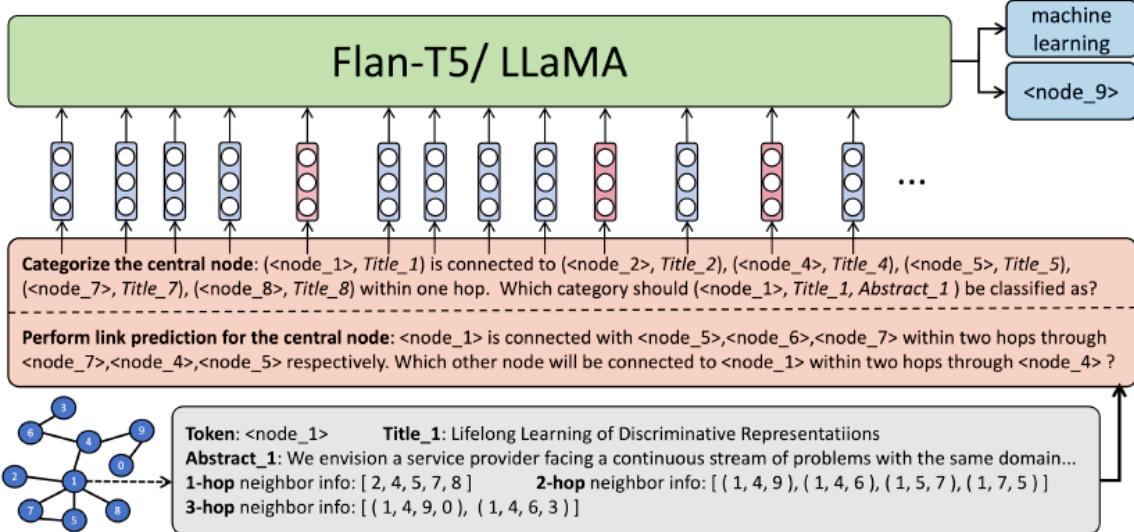


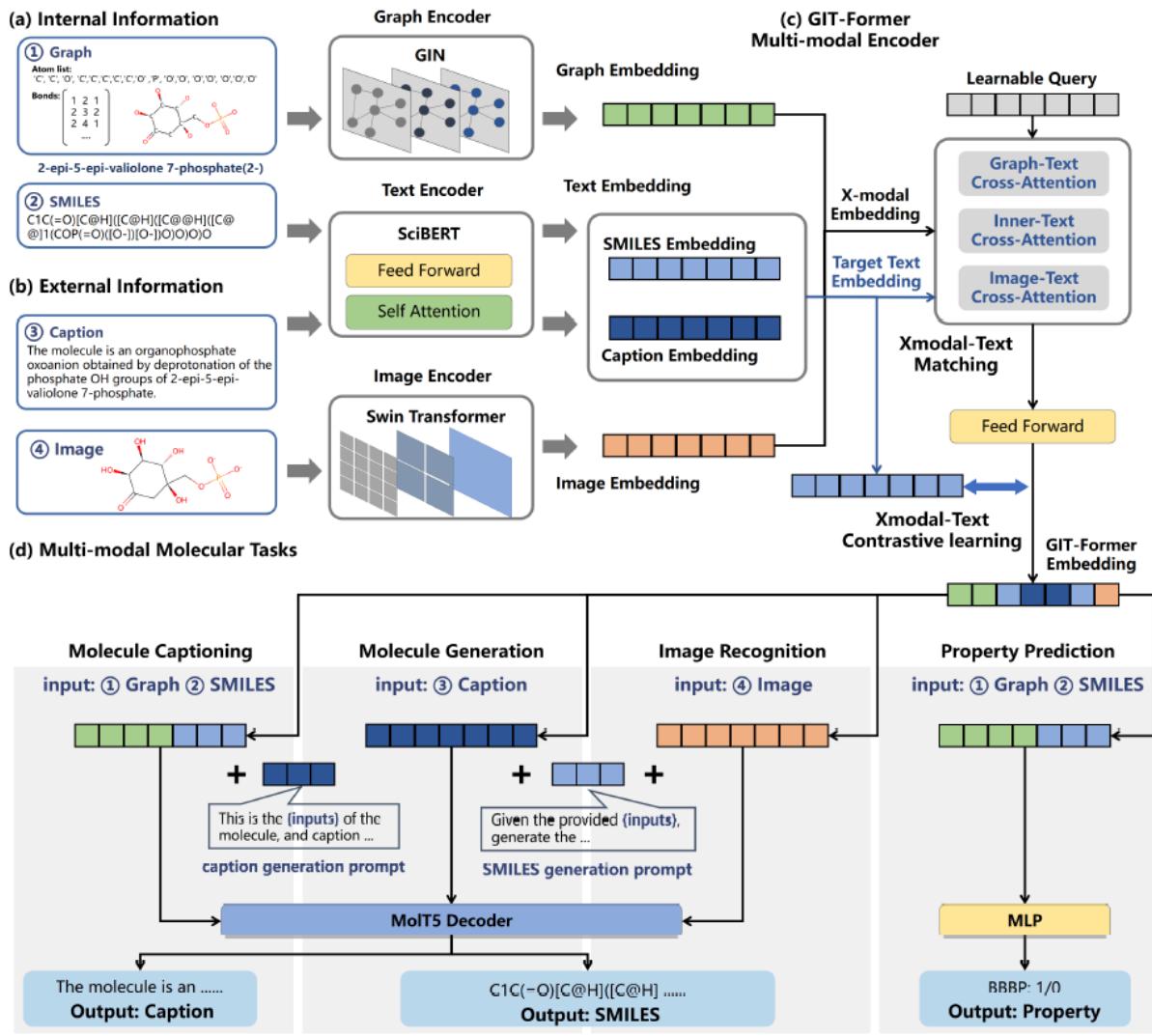
Figure 2: Illustration of InstructGLM. We use graph prompts to describe each node's multi-hop connectivity and meta features in a scalable mini-batch manner, conveying graph structure concisely and intuitively by pure natural language for learning. Subsequently, we instruct LLMs to generate responses for various graph tasks in a unified language modeling pipeline. We also expand the LLM's vocabulary by creating a new and unique token for each node. More specifically, we set the graph's inherent node feature vectors (e.g. BoW, OGB) as the embedding for these new tokens (depicted as red vectors in the figure) and employ the LLM's pre-trained embedding (depicted as blue vectors in the figure) for natural language tokens.

35.

P. Liu, Y. Ren, J. Tao, and Z. Ren, "Git-mol: A multi-modal large language model for molecular science with graph, image, and text," *Computers in Biology and Medicine*, vol. 171, p. 108073, Mar. 2024. [Online]. Available: <http://dx.doi.org/10.1016/j.combiomed.2024.108073>

Title: *GIT-Mol, a multi-modal large language model designed to integrate graph, image, and text data*

The article proposes GIT-Mol, a multi-modal large language model for molecular science, which integrates graph, image, and text data through a novel architecture called GIT-Former. The methodology involves aligning these modalities into a unified latent space using cross-attention mechanisms, where queries from one modality (e.g., text) interact with keys/values from others (e.g., graph or image). Pre-training employs Xmodal-Text Matching (XTM) and Xmodal-Text Contrastive Learning (XTC) to align representations, leveraging frozen encoders (Swin Transformer for images, GIN for graphs, and SciBERT/MoITS for text). During fine-tuning, prompt learning guides task-specific outputs, enabling "any-to-language" translation (e.g., image→SMILES, graph→caption) and molecular property prediction via MLP layers. The workflow spans data collection (4.8M PubChem compounds and ChEBI-20 pairs), rigorous preprocessing (cleaning, RDKit validation), pre-training for modality alignment, and task-specific fine-tuning for molecule captioning, generation, image recognition, and property prediction, achieving improved accuracy and validity over single-modality baselines.



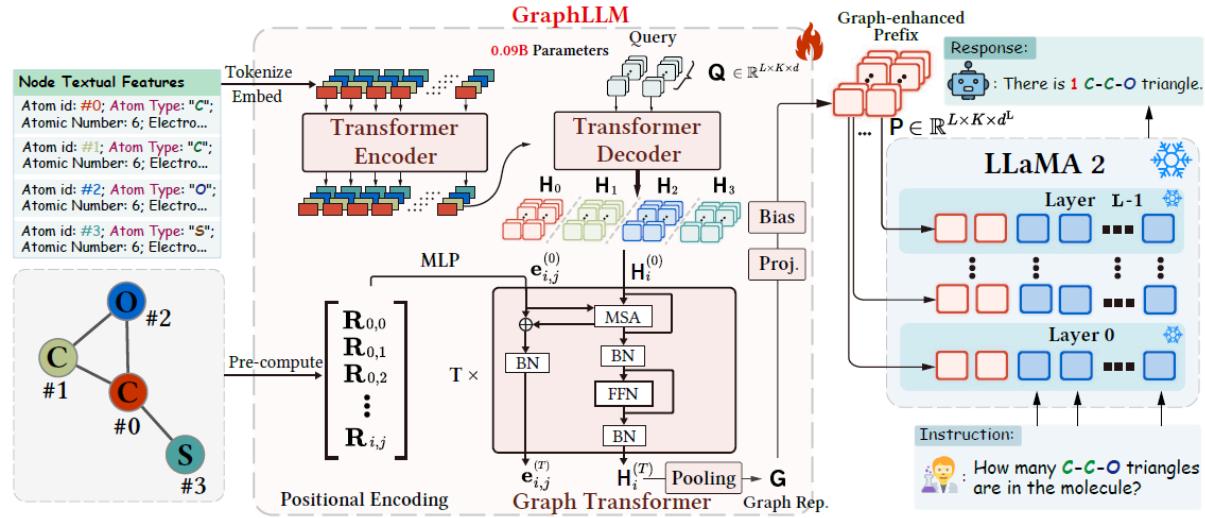
36.

Z. Chai, T. Zhang, L. Wu, K. Han, X. Hu, X. Huang, and Y. Yang, “GraphLLM: Boosting graph reasoning ability of large language model,” 2023. [Online]. Available: <https://arxiv.org/abs/2310.05845>

Title: GRAPHLLM: BOOSTING GRAPH REASONING ABILITY OF LARGE LANGUAGE MODEL

The article introduces GraphLLM, a method to enhance Large Language Models' (LLMs) graph reasoning by integrating them with graph learning models. The workflow involves three steps: (1) Node Understanding, where a transformer encoder-decoder extracts semantic features from

node textual descriptions (e.g., atom properties); (2) Structure Understanding, where a graph transformer models relationships between nodes using positional encodings and attention mechanisms to capture graph topology; and (3) Graph-enhanced Prefix Tuning, which condenses the graph representation into a fixed-length prefix prepended to the LLM’s attention layers. This prefix allows the LLM to reason over the graph without lengthy text descriptions. The entire system is trained end-to-end, enabling the LLM to synergize with the graph transformer for tasks like shortest-path computation or substructure counting, while drastically reducing context length and improving efficiency.



37.

Z. Liu, S. Li, Y. Luo, H. Fei, Y. Cao, K. Kawaguchi, X. Wang, and T.-S. Chua, “Molca: Molecular graph-language modeling with cross-modal projector and uni-modal adapter,” 2024. [Online]. Available: <https://arxiv.org/abs/2310.12798>

Title: MolCA: Molecular Graph-Language Modeling with Cross-Modal Projector and Uni-Modal Adapter

The article introduces MolCA, a method that integrates 2D molecular graphs with text generation through a three-stage framework. First, a graph encoder (GINE), pretrained on ZINC15 via contrastive learning, extracts structural features from molecules. Then, a cross-modal projector (Q-Former)—initialized from Sci-BERT—aligns graph and text spaces using learnable query tokens and cross-attention to interface 2D graph features with the language model (LM) input. The training involves three stages: Pretrain Stage 1 jointly trains the Q-Former and graph encoder using cross-modal tasks like contrastive learning, matching, and captioning to align

molecule-text pairs; Pretrain Stage 2 freezes the LM (Galactica) and trains the Q-Former to produce graph-conditioned soft prompts, integrating 1D SMILES with 2D structures to combine textual and structural insights; finally, the Fine-Tune Stage adapts the LM to specific downstream tasks (e.g., IUPAC name prediction) using LoRA, which updates only low-rank adapter matrices to preserve the LM’s generative capacity while remaining parameter-efficient. This approach enables LMs to generate text conditioned on molecular structures without fine-tuning the full language model.

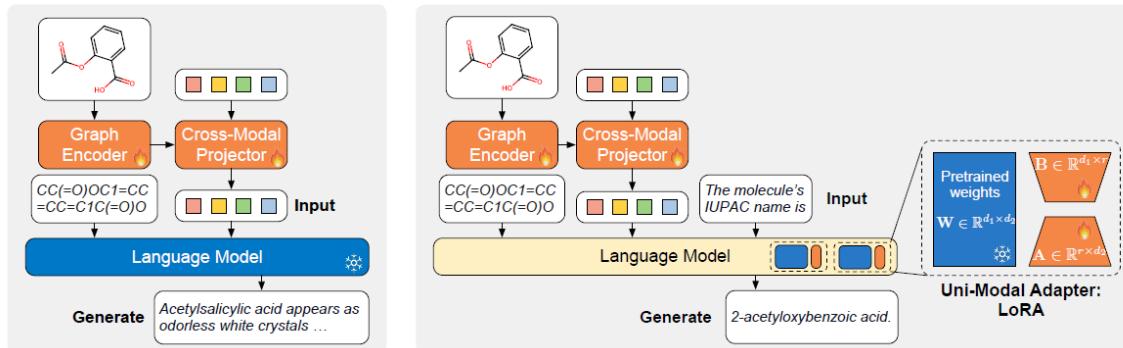
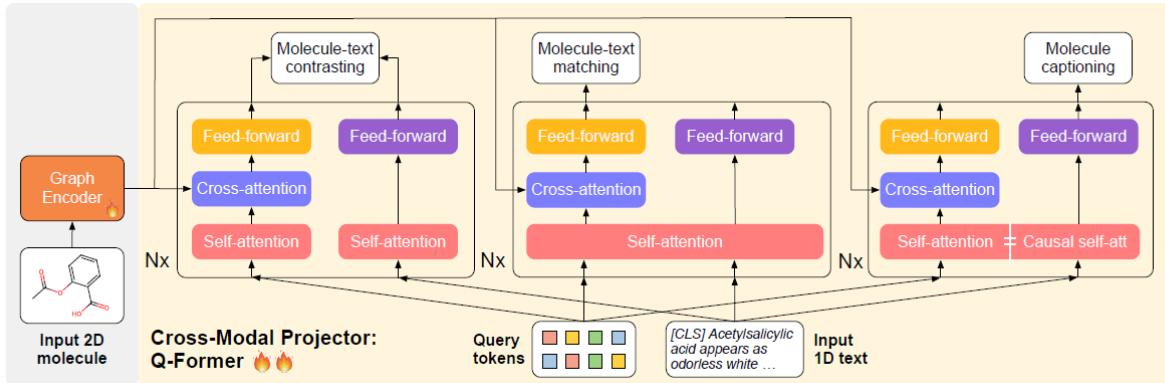


Figure 4: MolCA’s pretrain stage 2 by molecule captioning.

Figure 5: MolCA’s fine-tune stage for molecule-to-text generation. The example shows the prediction of a molecule’s IUPAC name.

38.

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,” 2024. [Online]. Available: <https://arxiv.org/abs/2310.13023>

Title: *GraphGPT: Graph Instruction Tuning for Large Language Models*

The GraphGPT framework introduces a novel dual-stage instruction tuning methodology to integrate graph structural knowledge into large language models (LLMs) without modifying their parameters. In the first stage, *self-supervised instruction tuning*, a graph encoder transforms graph structures into embeddings, which are aligned with textual node descriptions using contrastive learning and a lightweight projection layer. This enables the LLM to associate structural graph tokens with language tokens. In the second stage, *task-specific instruction tuning*, only the projector is fine-tuned to adapt the model for downstream graph tasks like node classification and link prediction using instructional prompts. To further improve reasoning, GraphGPT incorporates *Chain-of-Thought distillation* by transferring step-by-step reasoning capabilities from a stronger model (e.g., GPT-3.5). This architecture allows GraphGPT to efficiently and effectively reason over graph data in both supervised and zero-shot scenarios.

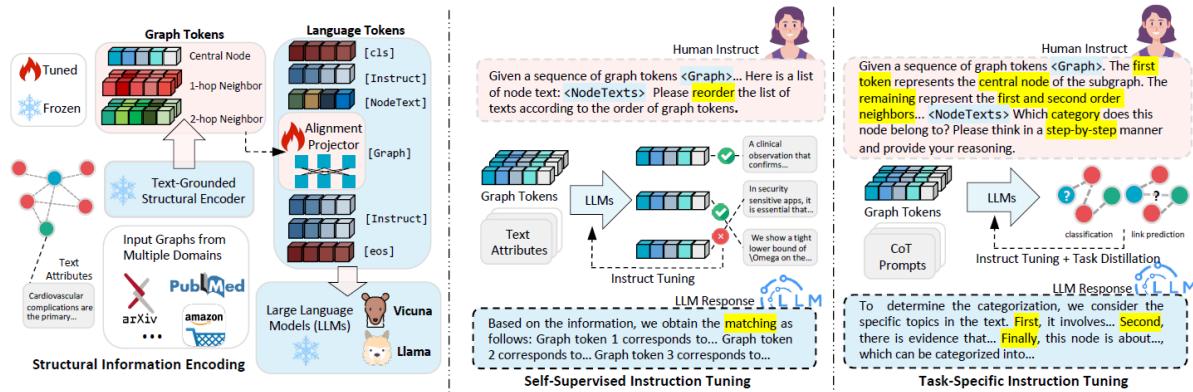


Figure 2: The overall architecture of our proposed GraphGPT with graph instruction tuning paradigm.

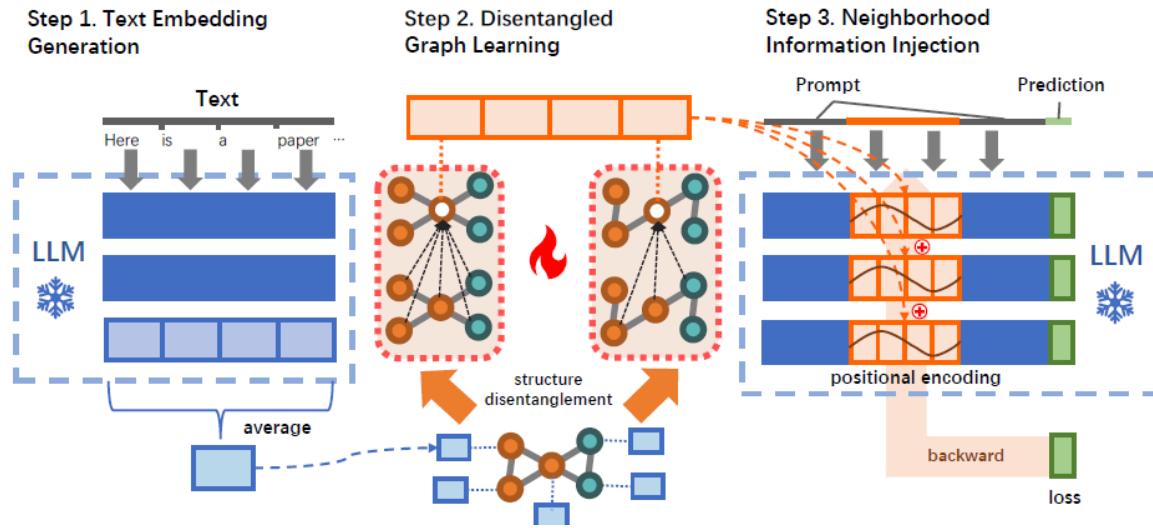
39.

Y. Qin, X. Wang, Z. Zhang, and W. Zhu, “Disentangled representation learning with large language models for text-attributed graphs,” 2024. [Online]. Available: <https://arxiv.org/abs/2310.18152>

Title: *Disentangled Representation Learning with Large Language Models for Text-Attributed Graphs*

The paper proposes Disentangled Graph-Text Learner (DGTL), a framework that enhances large language models (LLMs) for text-attributed graph (TAG) tasks by incorporating graph structure into the LLM's reasoning process without fine-tuning the LLM itself. The method begins by generating text embeddings for each node using a frozen, pretrained LLM. Then, multiple disentangled GNN layers are applied in parallel to capture diverse neighborhood structures, each learning different structural aspects via soft edge weights. These disentangled features are injected into the downstream LLM at reserved token positions and across all transformer layers, enabling the LLM to jointly reason over textual and structural information. The training updates

only the GNN layers, ensuring efficiency and flexibility, while the LLM handles the final prediction via prompt-based generation, often with interpretable natural language explanations.

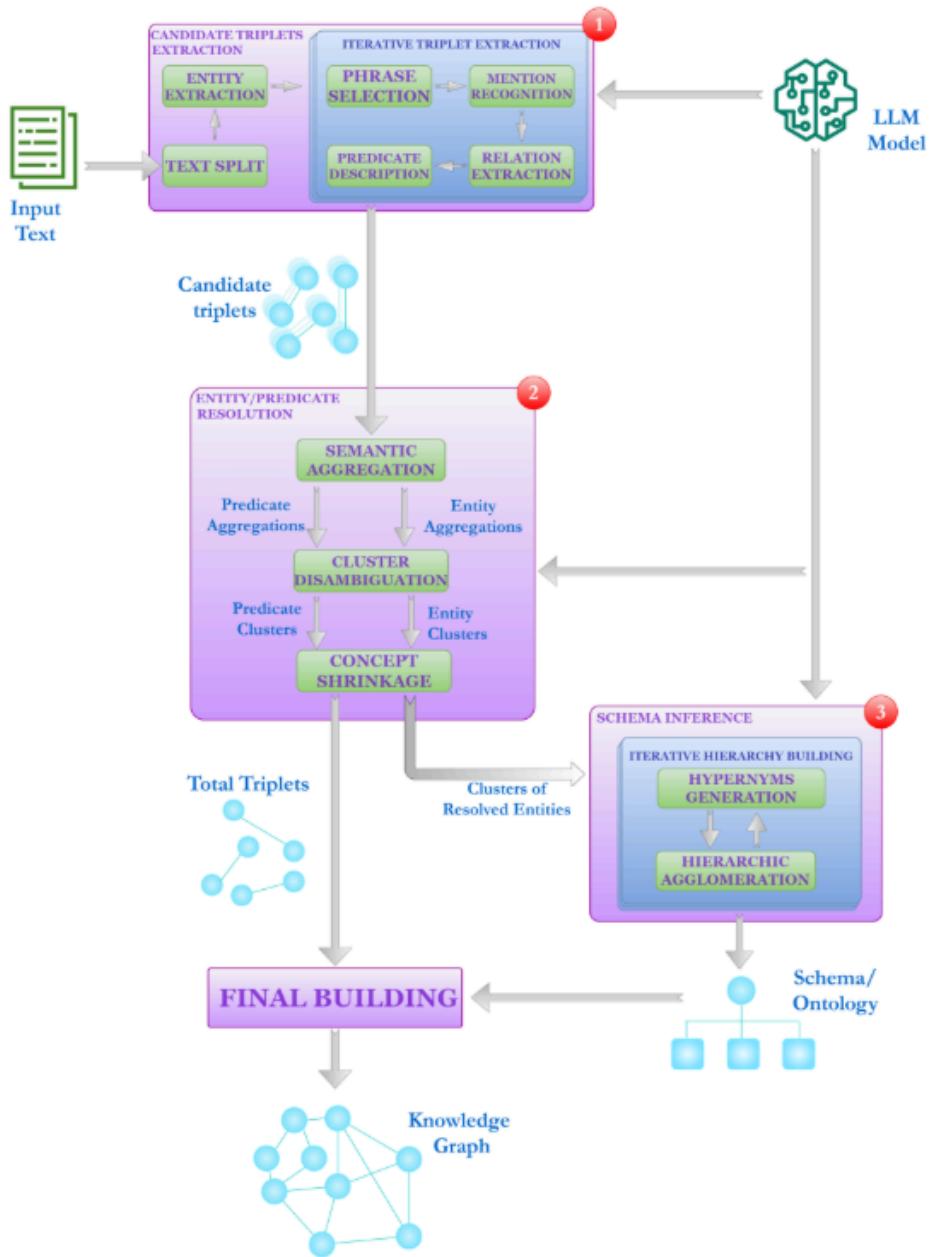


40.

S. Carta, A. Giuliani, L. Piano, A. S. Podda, L. Pompianu, and S. G. Tiddia, “Iterative zero-shot llm prompting for knowledge graph construction,” 2023. [Online]. Available: <https://arxiv.org/abs/2307.01128>

Title: *ITERATIVE ZERO-SHOT LLM PROMPTING FOR KNOWLEDGE GRAPH CONSTRUCTION*

The methodology proposed in this article revolves around an iterative, zero-shot prompting strategy leveraging GPT-3.5 to automate knowledge graph (KG) construction from unstructured text. The workflow begins with extracting relevant text excerpts for each target entity to reduce complexity and context loss, followed by generating candidate entities, types, descriptions, and triplets through well-designed, stage-specific prompts. This process involves multiple iterations, where extracted elements are refined by exploiting the semantic richness provided by the language model, without relying on external knowledge bases or human intervention. The approach emphasizes entity and predicate resolution, semantic enrichment, and schema inference to semantically "semantify" the extracted components, ultimately producing a structured, formalized KG in a scalable and knowledge-base agnostic manner.



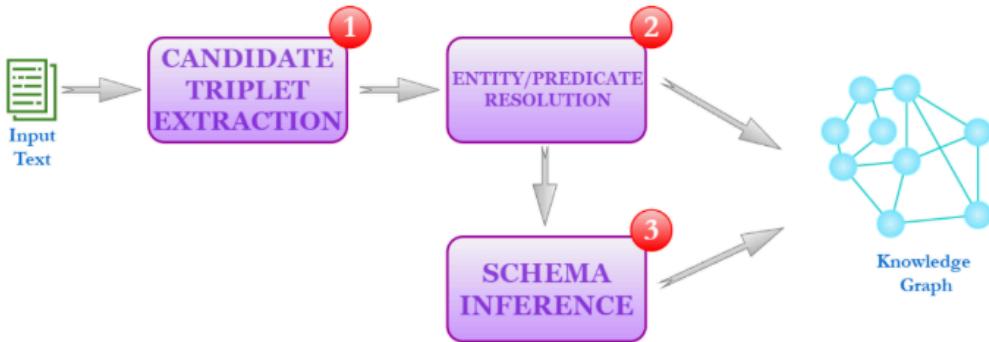


Figure 1: High-level architecture

41.

C. Shi, Y. Tao, H. Zhang, L. Wang, S. Du, Y. Shen, and Y. Shen, “Deep semantic graph learning via llm based node enhancement,” 2025. [Online]. Available: <https://arxiv.org/abs/2502.07982>

Title: *Deep Semantic Graph Learning via LLM based Node Enhancement*

The paper proposes a novel graph learning framework that integrates Large Language Models (LLMs) with Graph Neural Networks (GNNs), specifically through feature-level node enhancement. The workflow begins by extracting rich semantic embeddings for each node's textual attributes using various LLMs (e.g., SBERT, ADA, Google PaLM). These embeddings are then used as input features for graph-based models, including Graph Convolutional Networks (GCNs) and Graph Transformers. The framework evaluates different combinations of LLM-derived features and graph architectures across standard datasets (Cora and PubMed), using both low- and high-label scenarios. Models are trained using standardized splits and hyperparameters, with performance measured primarily via classification accuracy. Experimental results show that LLM-enhanced features significantly improve node classification, especially when paired with Graph Transformers under high-label conditions. This methodology highlights the importance of combining deep semantic understanding from LLMs with structural learning capabilities of GNNs.

42.

Q. Li, C. Huang, S. Li, Y. Xiang, D. Xiong, and W. Lei, "GraphOTTER: Evolving llm-based graph reasoning for complex table question answering," 2024. [Online]. Available: <https://arxiv.org/abs/2412.01230>

Title: *GraphOTTER: Evolving LLM-based Graph Reasoning for Complex Table Question Answering*

The GraphOTTER method introduces a novel, explicit reasoning workflow for complex table question answering (QA) by combining graph-based representations with large language models (LLMs). First, the input table is transformed into an undirected graph where both header and data cells are represented as nodes, and edges link cells in the same row or column. Rather than relying on implicit reasoning over entire tables, GraphOTTER initiates reasoning by selecting a small set of relevant nodes (up to 8) to form a "reasoning trace"—a subgraph that guides the process. Using a predefined set of reasoning actions (e.g., VisitNode, GetAllNeighbours, GetSharedNeighbours, AnswerQuestion), the LLM is prompted in a step-by-step manner to navigate the graph, update the trace, and isolate relevant data. The answer is generated only after the LLM determines sufficient information has been gathered. This approach filters out irrelevant content, builds a transparent reasoning path, and improves QA performance, especially on complex, real-world tables.

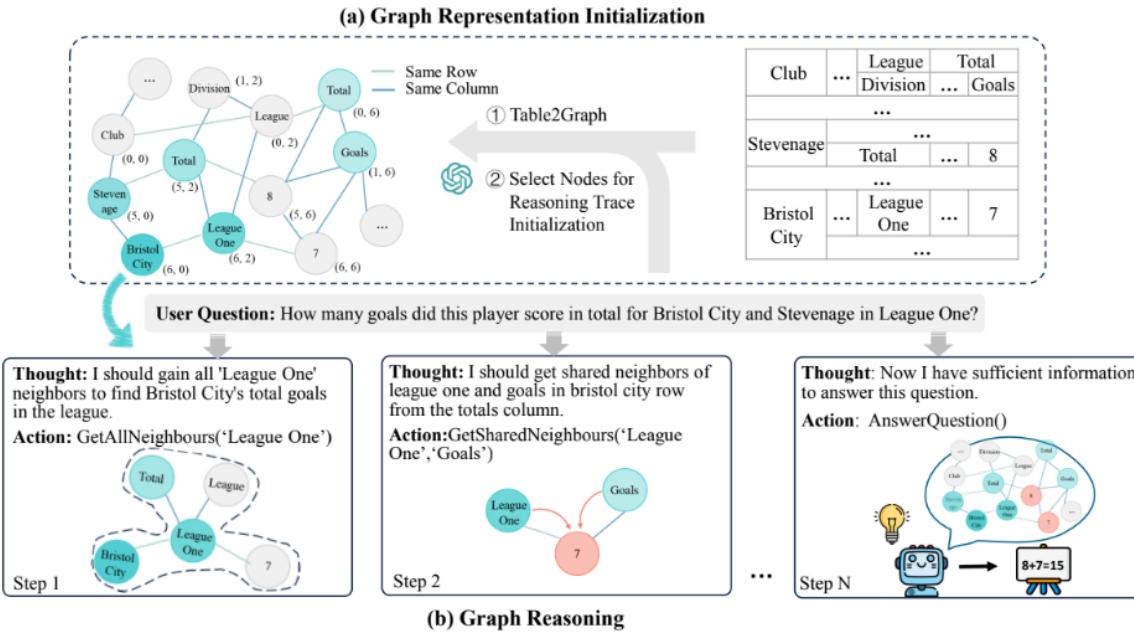


Figure 2: Overview of GraphOTTER. It first transforms the table into a graph representation (For clarity, this graph only shows some of the edges between nodes in the same row/column). Then, it establishes a step-by-step reasoning process on this graph, guided by a reasoning trace and a set of pre-defined intermediate actions (Note that the nodes in the reasoning trace are colored). These actions facilitate navigation and reasoning within the graph structure, leading to a more explicit and efficient approach for complex table QA.

43.

Y. Ji, C. Liu, X. Chen, Y. Ding, D. Luo, M. Li, W. Lin, and H. Lu, “Nt-llm: A novel node tokenizer for integrating graph structure into large language models,” 2024. [Online]. Available: <https://arxiv.org/abs/2410.10743>

Title: *NT-LLM: A Novel Node Tokenizer for Integrating Graph Structure into Large Language Models*

The NT-LLM framework introduces a novel method for integrating graph structures into Large Language Models (LLMs) via a custom node tokenizer. The methodology centers on encoding graph topology through Node Position Encoding—selecting a small set of anchor nodes and representing each node by its relative distances to these anchors. These distance vectors are then mapped into Euclidean space through a pretraining phase, preserving partial ordering and spatial relationships. These position embeddings, combined with textual attributes, are fed into the LLM using prompt tuning and further refined through Low-Rank Adaptation (LoRA) for task-specific tuning. This approach enables LLMs to capture both semantic and structural information in graph data without relying on external GNNs or graph-to-text conversions. The workflow is evaluated across multiple graph-related tasks, showing superior performance to existing GNN, LLM-only, and hybrid methods.

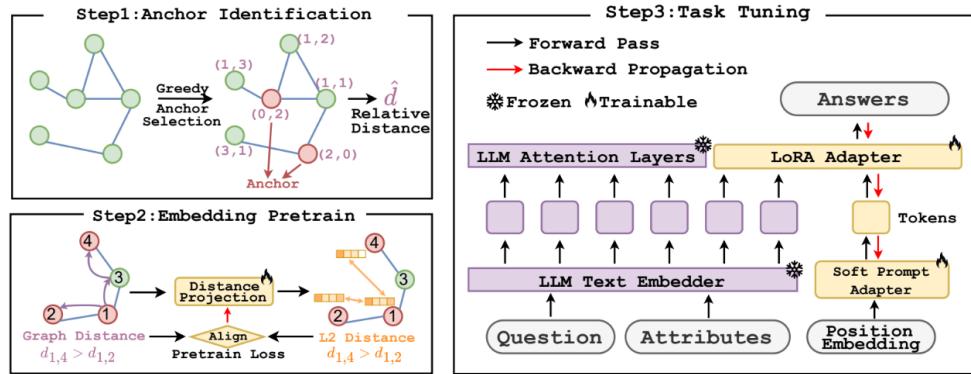


Figure 2: Overview of our proposed NT-LLM approach. Our method addresses graph-related tasks through three main steps: (1) **Anchor Identification**: We select key nodes as anchors with a greedy algorithm and compute relative distances between nodes and these anchors to encode the graph structure. (2) **Embedding Pretraining**: The relative distances are projected into a continuous Euclidean space while preserving the partial ordering of node distances. (3) **Task Tuning**: We integrate the pretrained embeddings with a Large Language Model (LLM) using a soft prompt adapter for position embeddings and LoRA for task-specific fine-tuning of the LLM.

44.

Q. Cai, Z. Wang, S. Diao, J. Kwok, and Y. Song, “Codegraph: Enhancing graph reasoning of llms with code,” 2024. [Online]. Available: <https://arxiv.org/abs/2408.13863>

Title: CodeGraph: Enhancing Graph Reasoning of LLMs with Code

The methodology of the *CodeGraph* framework revolves around enhancing LLMs' performance on basic graph reasoning tasks by delegating computation to executable Python code rather than relying solely on natural language reasoning. The process begins with encoding graph-structured data into natural language using one of six predefined encoding functions (e.g., adjacency, friendship). In a few-shot learning setting, the LLM is prompted with exemplars consisting of graph-based questions and corresponding Python code that solve specific tasks (e.g., edge existence, node degree). Given a new graph question, the LLM generates a Python program encapsulated between # CODE START and # CODE END, which is then executed externally to obtain the final answer. This program-aided reasoning approach improves arithmetic accuracy, mitigates sensitivity to textual prompts, and provides interpretability. CodeGraph is evaluated across diverse tasks and graph structures using multiple LLMs (e.g., GPT-3.5 Turbo, Llama3-70B), consistently showing superior robustness and generalization over baselines.

45.

Y. Li, Y. Yang, J. Zhu, H. Chen, and H. Wang, “Llm-empowered few-shot node classification on incomplete graphs with real node degrees,” in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, ser. CIKM ’24. New York, NY, USA: Association for Computing Machinery, 2024, p. 1306–1315. [Online]. Available: <https://doi.org/10.1145/3627673.3679861>

Title: *LLM-Empowered Few-Shot Node Classification on Incomplete Graphs with Real Node Degrees*

The methodology of the LLMDGCN framework addresses few-shot node classification on incomplete graphs using real node degrees by combining large language models (LLMs) with graph neural networks (GCNs) in a two-module workflow. First, in the LLM generation module, LLMs are used to synthesize virtual nodes and generate pseudo-labels with confidence scores to fill in missing nodes and labels. These are then embedded using a pre-trained language model (PLM). In the iterative prediction module, a GCN is trained using the augmented data to produce updated pseudo-labels. Simultaneously, an edge predictor restores missing edges based on the

real node degrees and an inter-category probability matrix that estimates connection likelihoods between classes. The refined graph is fed back into the GCN, and this process iterates until convergence. This iterative interplay progressively enhances the training data and graph structure, yielding improved node classification accuracy even in zero- or few-shot scenarios.

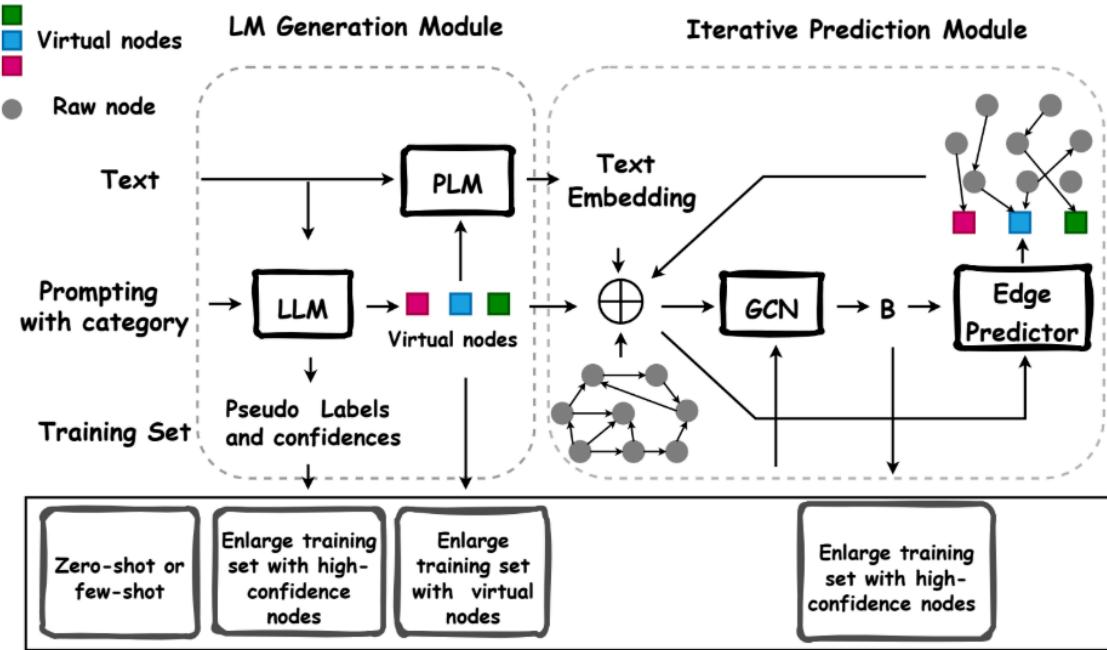


Figure 2: The overall architecture of LLMDGCN.

46.

- T. Feng, Y. Shen, and J. You, “Graphrouter: A graph-based router for llm selections,” 2025. [Online]. Available: <https://arxiv.org/abs/2410.03834>

Title: GRAPHROUTER: A GRAPH-BASED ROUTER FOR LLM SELECTIONS

The methodology of GraphRouter introduces an inductive graph-based framework for selecting the most suitable Large Language Model (LLM) for a given user query by modeling contextual interactions among tasks, queries, and LLMs. It constructs a heterogeneous graph where tasks, queries, and LLMs are represented as nodes, and their relationships—such as performance and cost of responses—are encoded as edge features. Initial embeddings for nodes are generated using descriptions from LLMs (like GPT-4o) and encoded with a pre-trained language model (e.g., BERT). A heterogeneous GNN is then applied to iteratively update node embeddings and predict the best LLM for each query by treating the selection task as an edge prediction problem.

The framework supports generalization to new LLMs without retraining, by using few-shot data and inductively derived embeddings. This enables GraphRouter to efficiently balance performance and cost across diverse tasks and evolving LLM pools.

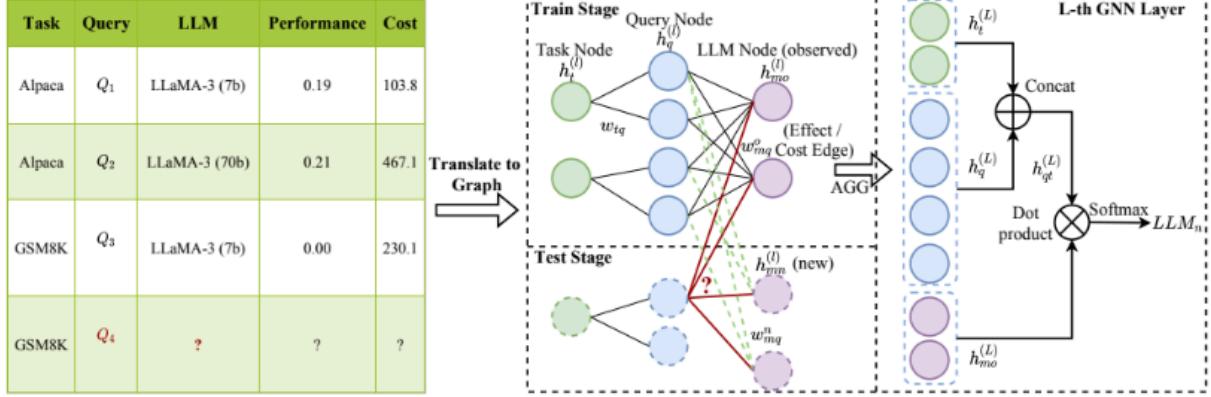


Figure 5: **Overview of GraphRouter methodology.** GraphRouter first converts the interaction data among tasks, queries, and LLMs into a graph. Specifically, as illustrated on the right side, tasks, queries, and LLMs from the left table are represented as task nodes, query nodes, and LLM nodes, respectively. Moreover, their relationships derived from the interaction data are modeled as edge features. With this structure, we leverage a GNN to embed both node and edge features, ultimately producing the probability distribution of the selected LLM.

47.

E. Du, X. Li, T. Jin, Z. Zhang, R.-H. Li, and G. Wang, “Graphmaster: Automated graph synthesis via llm agents in data-limited environments,” 2025. [Online]. Available: <https://arxiv.org/abs/2504.00711>

Title: GRAPHMASTER: AUTOMATED GRAPH SYNTHESIS VIA LLM AGENTS IN DATA-LIMITED ENVIRONMENTS

The methodology of the *GraphMaster* framework is centered around a hierarchical, retrieval-augmented generation (RAG)-based multi-agent system designed for synthesizing high-quality text-attributed graphs in data-limited environments. The process begins with the Manager Agent, which analyzes the current graph and dynamically selects an enhancement mode (semantic or topological) to formulate a synthesis query. This query is passed to the Perception Agent, which overcomes LLM context window limitations by extracting a representative, semantically and structurally informative subgraph using community detection and adaptive sampling. The Enhancement Agent then generates new nodes and edges conditioned on the retrieved subgraph, ensuring semantic coherence and structural integrity through mode-specific generation mechanisms. Finally, the Evaluation Agent assesses the quality of the generated content using both semantic and structural criteria, filtering low-quality additions and providing

feedback for iterative refinement. This closed-loop process continues until convergence, resulting in an enhanced graph that merges the original structure with high-quality synthetic elements. Overall, this workflow integrates modular reasoning, targeted generation, and iterative quality control to robustly synthesize graphs suitable for training graph foundation models under data scarcity.



Figure 1: GraphMaster: A hierarchical multi-agent framework for text-attributed graph synthesis.

48.

Y. Hu, R. Lei, X. Huang, Z. Wei, and Y. Liu, “Scalable and accurate graph reasoning with llm-based multi-agents,” 2024. [Online]. Available: <https://arxiv.org/abs/2410.05130>

Title: SCALABLE AND ACCURATE GRAPH REASONING WITH LLM-BASED MULTI-AGENTS

The GraphAgent-Reasoner (GAR) framework introduces a fine-tuning-free, multi-agent methodology for scalable and accurate graph reasoning using large language models (LLMs). Its core idea is to emulate distributed graph computation by assigning each node in a graph to a separate reasoning agent. The system is coordinated by a central Master LLM, which parses a graph problem (given as a natural language description and task query), constructs the agent network based on graph structure, and designs or retrieves a suitable distributed algorithm composed of six modular components: state, message, initialization, send, update, and termination. Agents then execute the algorithm in parallel by exchanging messages and updating their states iteratively. Once the termination condition is met, agents send their final states to the Master LLM, which aggregates the results and returns the final answer. This approach enables explicit reasoning paths, reduces the burden on any single LLM, and allows scalability to graphs with over 1,000 nodes—all without model fine-tuning.

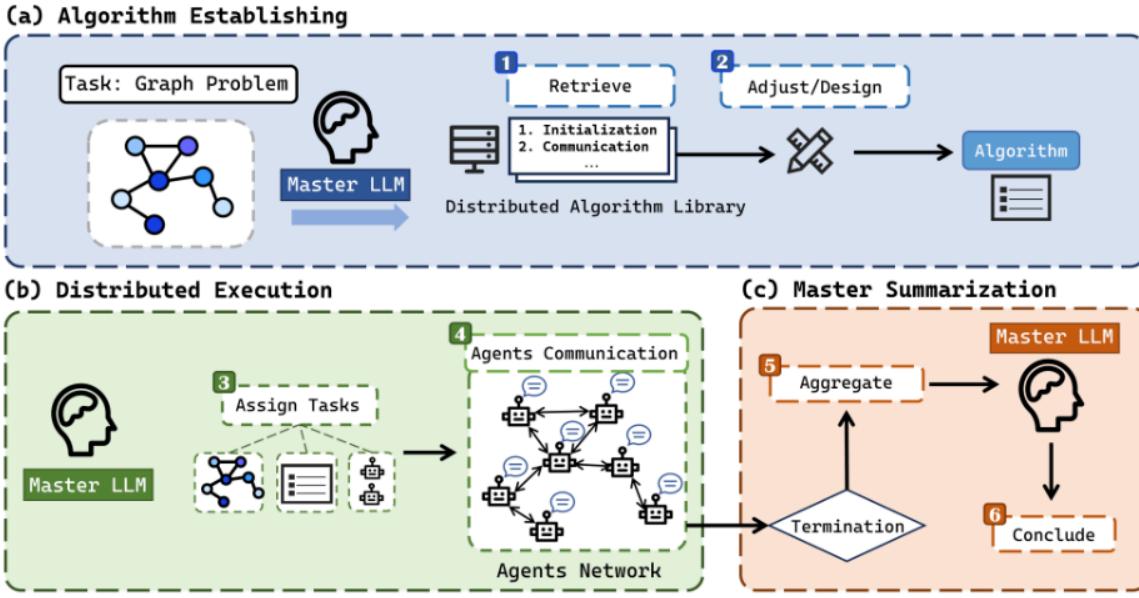


Figure 3: The framework of GraphAgent-Reasoner. Given a graph problem, the Master LLM will first construct agents network according to graph structures. It then sequentially performs Algorithm Establishing, Distributed Execution and Master Summarization, as detailed in this section.

49.

X. Zhu, H. Xue, Z. Zhao, W. Xu, J. Huang, M. Guo, Q. Wang, K. Zhou, and Y. Zhang, “Llm as gnn: Graph vocabulary learning for text-attributed graph foundation models,” 2025. [Online]. Available: <https://arxiv.org/abs/2503.03313>

Title: LLM as GNN: Graph Vocabulary Learning for Text-Attributed Graph Foundation Models

The PromptGFM framework introduces a novel methodology for building a Graph Foundation Model (GFM) for text-attributed graphs (TAGs) by operating large language models (LLMs) as graph neural networks (GNNs) through prompt-based learning. The approach consists of two key components. First, the Graph Understanding Module prompts LLMs to replicate the message-passing paradigm of GNNs within the text space. It initializes node features using their textual descriptions, samples one-hop neighbors, and iteratively performs aggregation and update operations via prompt templates to refine node representations—mimicking GNN behavior but producing compact textual outputs instead of embeddings. Second, the Graph Inference Module constructs a language-based graph vocabulary, converting each node into a sequence of interpretable tokens (language-based IDs). These IDs are embedded into natural language instructions tailored to graph tasks such as node classification or link prediction. The model is then fine-tuned across multiple graphs and tasks using these prompts in a unified

instruction-tuning setup. This enables robust semantic alignment, enhances transferability, and supports scalable reasoning over diverse and unseen graphs.

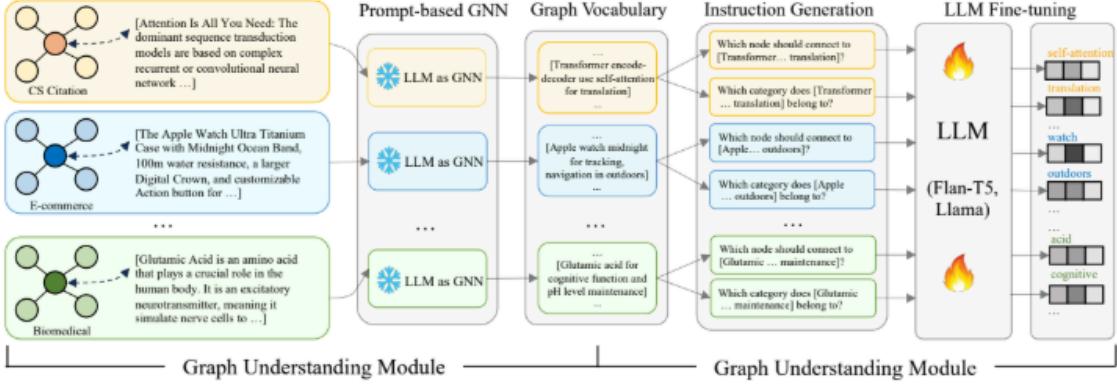


Figure 3: The pipeline of PromptGFM. (a) Graph Understanding Module: For arbitrary TAGs from different domains, prompt-based GNN replicates traditional embedding-based GNN workflow in the text space, generating compact node representations. (b) Graph Inference Module: We establish a unified graph vocabulary and extract language-based IDs to generate massive pure-language prompts, enabling LLM fine-tuning across graphs and tasks.

50.

G. Sun, Y. Wang, D. Niyato, J. Wang, X. Wang, H. V. Poor, and K. B. Letaief, “Large language model (llm)-enabled graphs in dynamic networking,” 2024. [Online]. Available: <https://arxiv.org/abs/2407.20840>

Title: Large Language Model (LLM)-enabled Graphs in Dynamic Networking

The methodology and workflow of the article involve designing a novel framework that integrates Large Language Models (LLMs) with graph-based structures to optimize dynamic networking. The proposed architecture consists of five layers: (1) the Input Layer receives multi-modal data from dynamic networks (e.g., UAVs and monitoring nodes); (2) the Graph-to-Text Layer extracts features and converts graph data into text using embeddings and prompt engineering; (3) the Decision Layer leverages LLMs (e.g., ChatGPT or Bard) for reasoning and prediction based on the textual input; (4) the Text-to-Graph Layer reconstructs graphs from LLM-generated text, embeds semantic and structural features, and uses Graph Neural Networks (GNNs) for learning and optimization; and (5) the Output Layer visualizes and verifies the final network configurations. A case study on UAV trajectory and communication resource allocation demonstrates the workflow’s efficacy using a two-stage optimization strategy: LLMs generate trajectories based on input features, and GNNs optimize communication resources accordingly, achieving better energy efficiency and task success.

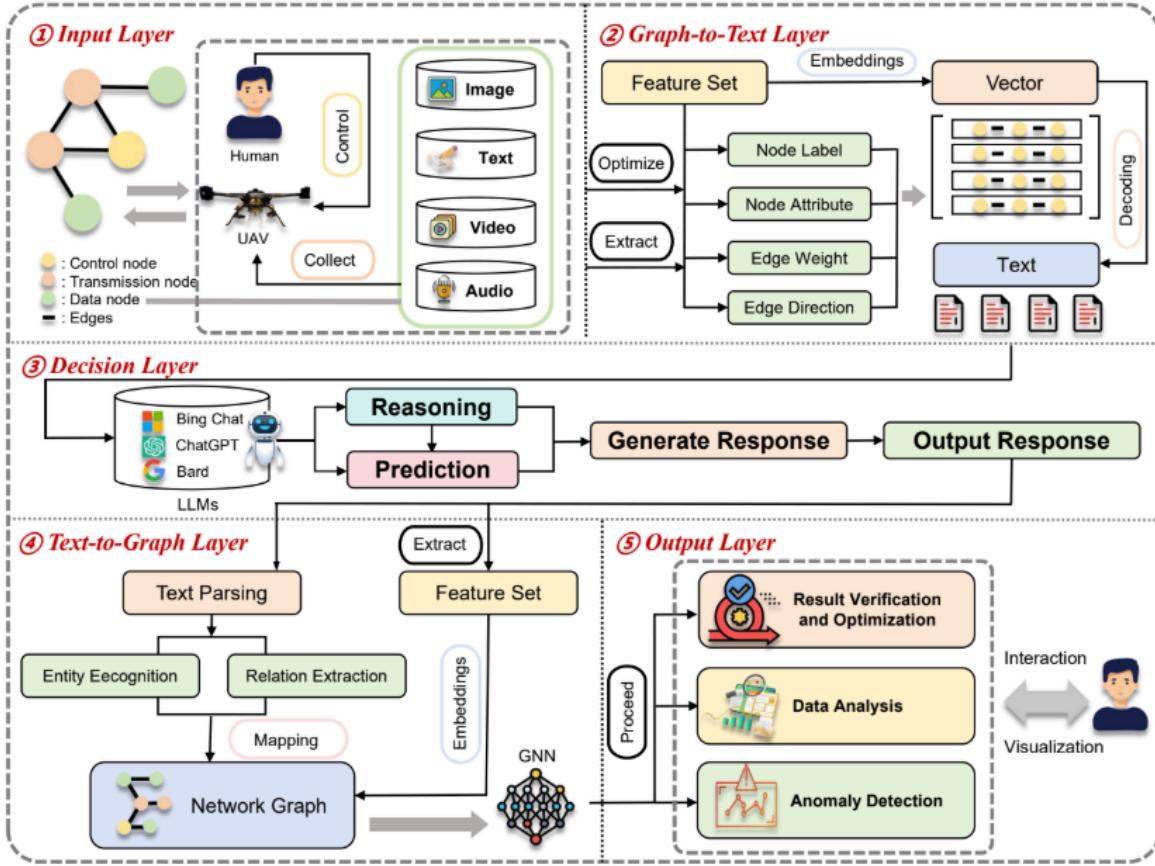


Fig. 4. The structure of the proposed LLM-enabled graphs framework. The framework is based on a layered architecture consisting of an input layer, a graph-to-text layer, a decision layer, a text-to-graph layer, and an output layer. The input layer receives requests related to a dynamic network graph. The graph-to-text layer employs prompt engineering to extract features from requests, and then converts them to text via embeddings. The decision layer utilizes a pluggable LLM to generate responses. The text-to-graph layer extracts features from the text generated by the LLM to construct the graph, which is then processed by a GNN. The output layer analyzes the generated results and interacts with the user.

51.

X. Wu, Y. Shen, F. Ge, C. Shan, Y. Jiao, X. Sun, and H. Cheng, “A comprehensive analysis on llm-based node classification algorithms,” 02 2025.

Title: *A Comprehensive Analysis on LLM-based Node Classification Algorithms*

The article presents a methodology that integrates Large Language Models (LLMs) with graph-based data structures to enhance dynamic network optimization. The proposed workflow consists of a multi-layer framework: the input layer collects multi-modal data (e.g., text, video, audio); the graph-to-text layer converts graph features into vectorized textual representations using prompt engineering; the decision layer employs an LLM (e.g., ChatGPT) to perform

reasoning and generate text outputs; the text-to-graph layer extracts key features from the LLM-generated text to reconstruct and enhance the graph structure using Graph Neural Networks (GNNs); and finally, the output layer verifies, analyzes, and visualizes the results for user interaction. This workflow is demonstrated through a case study on UAV networking, using a two-stage joint optimization: LLMs generate the UAV's trajectory, and GNNs optimize communication resource allocation, effectively minimizing energy consumption while ensuring task completion.

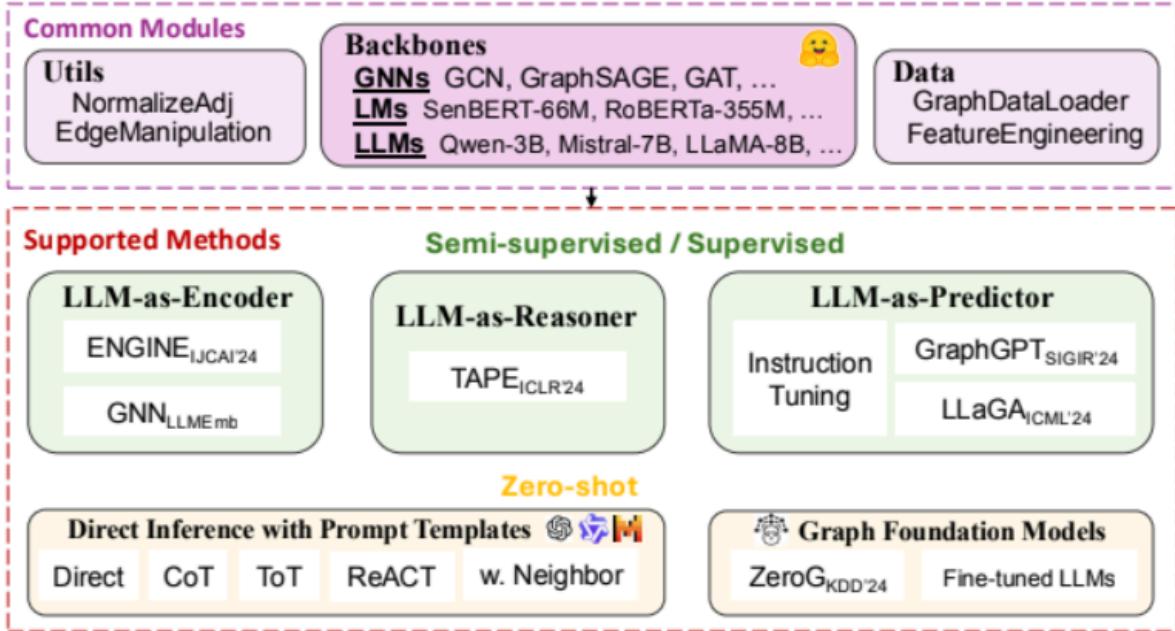


Figure 1: Overview of LLMNodeBed.

52.

T. Feng, Y. Sun, and J. You, “Grapheval: A lightweight graph-based llm framework for idea evaluation,” 2025. [Online]. Available: <https://arxiv.org/abs/2503.12600>

Title: GRAPHEVAL: A LIGHTWEIGHT GRAPH-BASED LLM FRAMEWORK FOR IDEA EVALUATION

The article employs a simulation-based methodology to evaluate the performance of a Multi-access Edge Computing (MEC) enabled Unmanned Aerial Vehicle (UAV) network for real-time target tracking. The workflow begins with modeling the system, which includes a UAV, MEC servers, and a moving target. The authors formulate the task offloading and UAV trajectory optimization problem as a mixed-integer non-linear programming (MINLP) problem. To solve

this, they propose a two-layer optimization approach: the first layer optimizes task offloading decisions using a deep reinforcement learning (DRL) algorithm, while the second layer optimizes the UAV's flight trajectory using a successive convex approximation (SCA) method. This layered approach enables adaptive and efficient tracking by balancing computational load and minimizing energy consumption. Simulation results validate the effectiveness of the proposed method in improving tracking accuracy and reducing system cost.

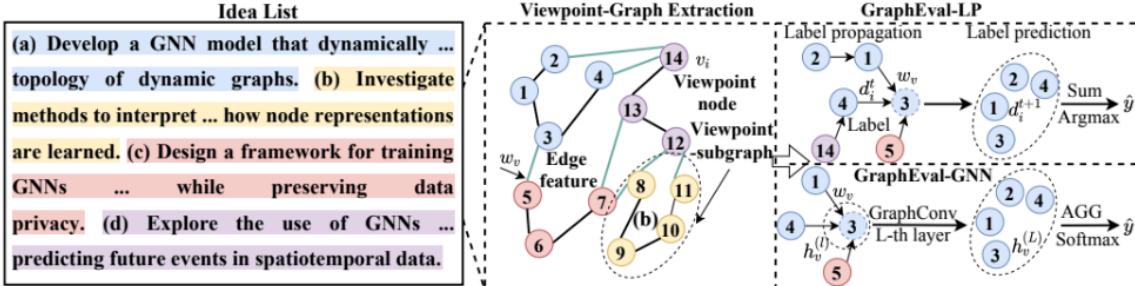


Figure 3: **Overview of GraphEval methodology.** GraphEval first transforms the ideas into a viewpoint-graph via Viewpoint-Graph Extraction, which contains multiple viewpoint-subgraphs, viewpoint-nodes, and edges between viewpoint-nodes. Then two lightweight GraphEval implementations named GraphEval-LP and GraphEval-GNN are employed to evaluate the ideas. Note that AGG denotes the acronym for aggregation function.

53.

W. Shang, X. Zhu, and X. Huang, “Path-llm: A shortest-path-based llm learning for unified graph representation,” 2024. [Online]. Available: <https://arxiv.org/abs/2408.05456>

Title: *Path-LLM: A Shortest-Path-based LLM Learning for Unified Graph Representation*

The Path-LLM framework introduces a novel methodology for unified graph representation learning by leveraging large language models (LLMs) and shortest-path-based features. The workflow begins with Phase I, where the model extracts *long-to-short shortest paths* (*L2SP*) to effectively capture both intra- and inter-group graph structures. These long shortest paths are sliced into shorter sub-paths, which are then textualized in Phase II by converting their structural information into natural language-like sequences using node and edge text attributes. In Phase III, these *L2SP*-based texts are used to pre-train an LLM in a self-supervised manner, allowing the model to learn semantic-rich graph structures by predicting the next node's text based on previous context. Finally, in Phase IV, node embeddings are generated by passing cleaned node texts through the trained Path-LLM and averaging token-level embeddings from its final layer.

These embeddings are used for downstream tasks like node classification, link prediction, and keyword search, where the structural and semantic strengths of L2SP help achieve superior performance with significantly fewer training paths.

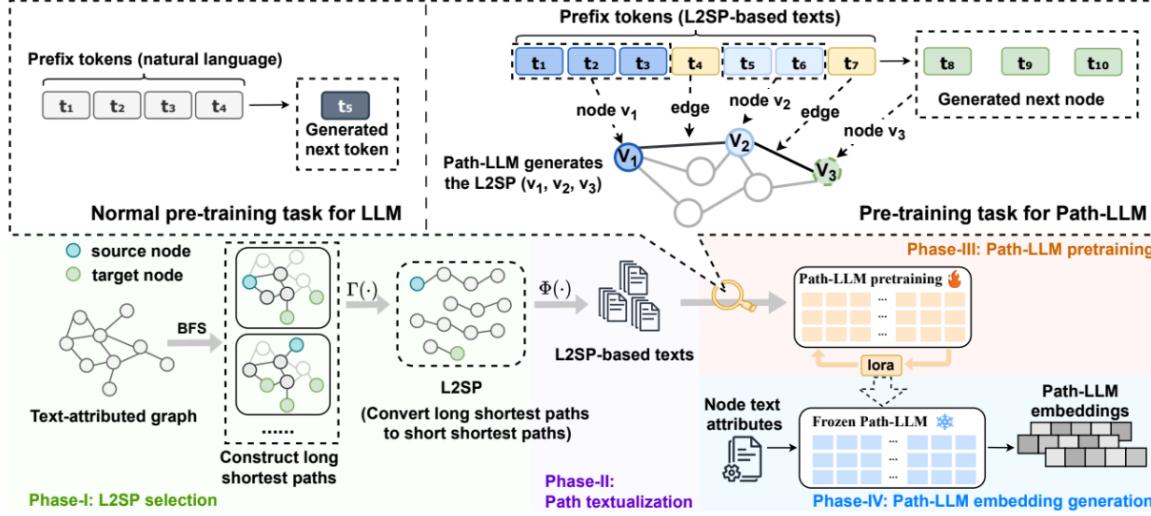


Figure 2: Our proposed Path-LLM framework involves four key components: (1) Long-to-short shortest path (L2SP) selection, converting sampled long shortest paths to L2SP-based shortest paths. (2) Path textualization, transforming L2SP-based shortest paths to L2SP-based texts. (3) Path-LLM pre-training, learning graph structures from L2SP-based texts. (4) Path-LLM embedding generation, deriving the final embeddings from the frozen Path-LLM.