Natural Language is All a Graph Needs

Anonymous ACL submission

Abstract

The emergence of large-scale pre-trained language models, such as ChatGPT, has revolutionized various research fields in artificial intelligence. Transformers-based large language models (LLMs) have gradually replaced CNNs and RNNs to unify fields of computer vision and natural language processing. Compared with the data that exists relatively independently such as images, videos or texts, graph is a type of data that contains rich structural and relational information. Meanwhile, natural language, as one of the most expressive mediums, excels in describing complex structures. However, existing work on incorporating graph learning problems into the generative language modeling framework remains very limited. As the importance of LLMs continues to grow, it becomes essential to explore whether LLMs can also replace GNNs as the foundation model for graphs. In this paper, we propose **Instruct-GLM** (**Instruct**ion-finetuned **G**raph **L**anguage **M**odel), systematically design highly scalable prompts based on natural language instructions, and use natural language to describe the geometric structure and node features of the graph for instruction tuning an LLM to perform learning and inference on graphs in a generative manner. Our method exceeds all competitive GNN baselines on ogbn-arxiv, Cora and PubMed datasets, which demonstrates the effectiveness of our method and sheds light on generative large language models as the foundation model for graph machine learning. Our code will be released once published.

1 Introduction

Before the advent of Transformers (Vaswani et al., 2017), various artificial intelligence domains with different inductive biases had diverse foundational model architectures. For instance, CNNs (He et al., 2016; Szegedy et al., 2016) were designed with considerations for spatial invariance in images, leading to superior performance in computer vision tasks (Deng et al., 2009; Lin et al., 2014). Memory-enhanced models like RNNs (Elman, 1990) and LSTM (Hochreiter and Schmidhuber, 1997; Cho et al., 2014) were widely used for handling sequential data such as natural language (Sarlin et al., 2020) and audio (Chen et al., 2021). Graph Neural Networks (GNNs) excel in capturing topological information by employing message passing and aggregation mechanisms, making them a preferred choice in the field of graph learning for a long time (Kipf and Welling, 2016; Veličković et al., 2017; Hamilton et al., 2017; Han et al., 2023a).

In recent years, the AI community has witnessed the emergence of numerous powerful pre-trained Large Language Models (LLMs) (Devlin et al., 2018; Raffel et al., 2020; Brown et al., 2020; Touvron et al., 2023; Ouyang et al., 2022), which are driving huge advancements and lead to the pursuit of possible Artificial General Intelligence (AGI) (Bubeck et al., 2023). Under this background, there is a trend towards unification in model architectures across different domains. Specifically, pre-trained Transformers have demonstrated remarkable performance on various modalities, such as images (Dosovitskiy et al., 2020) and videos (Arnab et al., 2021) in computer vision, text in natural language processing (Singh et al., 2021), structured data in graph machine learning (Ying et al., 2021), personalized data in recommender systems (Geng et al., 2022), decision sequences in reinforcement learning (Di Palo et al., 2023), and visual-text pairs in multimodal tasks (Radford et al., 2021). There has even been Transformers capable of handling twelve modalities (Zhang et al., 2023b).

Besides model architecture, the unification of processing method in handling multimodal data is also a significant trend worth attention. T5 (Raffel et al., 2020) established a text-to-text framework, unifying all NLP tasks as a sequence generation problem. Moreover, models like CLIP (Radford et al., 2021) utilize image-text pairs to accomplish

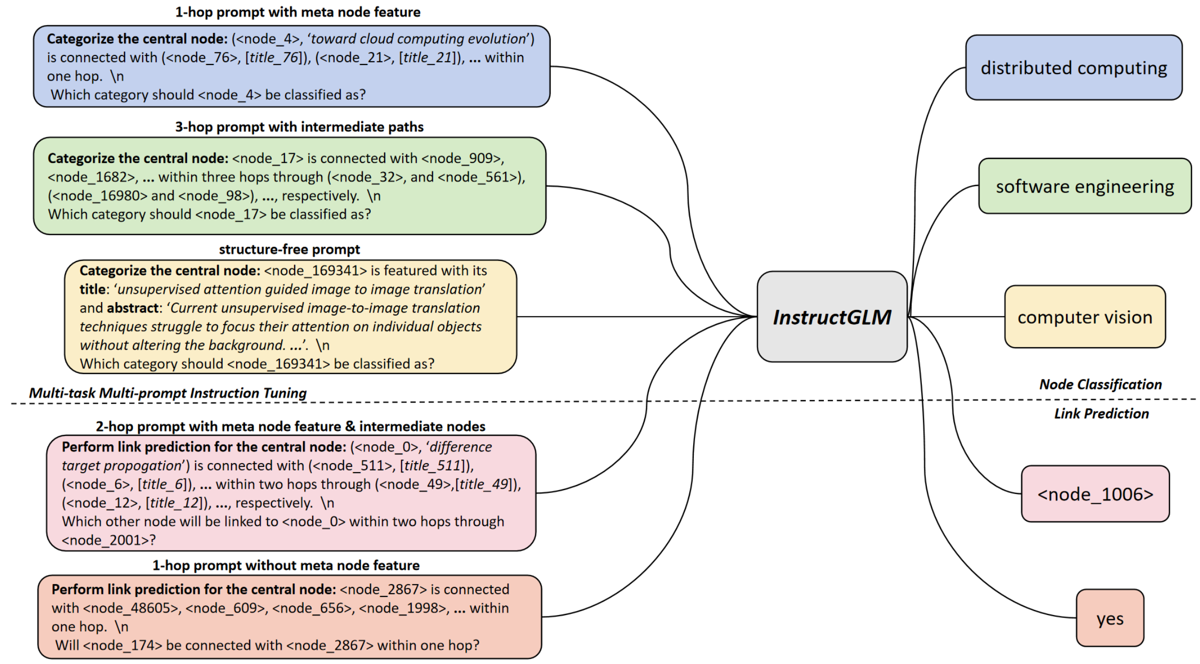

Figure 1: Illustration of the InstructGLM Framework. We fine-tune InstructGLM under a Multi-task Multi-prompt instruction tuning framework, enabling it to solve various graph machine learning tasks with the structure information purely described by natural language.

multimodal tasks with the images captioned by natural language. As for reinforcement learning, Di Palo et al. (2023) employs natural language to describe environmental states for the agent which successfully solves many reinforcement learning (RL) problems. P5 (Geng et al., 2022) further contributes to this trend by reformulating all personalized recommendation tasks as language modeling tasks via prompts. The aforementioned works collectively demonstrate that employing natural language for multimodal information representation has emerged as a prominent and promising trend.

However, in graph machine learning, such an exploration still remains limited. Existing methods that utilize large language models for graph tasks can be roughly categorized into two types: 1) Combining LLMs and GNNs, where the LLM acts as a feature extractor or data augmentation module to enhance the downstream GNNs' performance (He et al., 2023; Mavromatis et al., 2023; Zhao et al., 2023). Such kind of methods often require training multiple models, incurring significant computational overhead and tend to easily inherit drawbacks of GNNs such as over-smoothing (Cai and Wang, 2020). 2) Only relying on Transformers but necessitating novel designs of token embedding for nodes and edges (Kim et al., 2022) or creating complex graph attention module to learn structural information (Dwivedi and Bresson, 2020; Nguyen et al., 2022). This type of method demands local attention calculation on every node during each optimization step, leading to considerable computation costs and thus limiting each node's scope to only 1-hop neighbors. Meanwhile, the complex pipeline with special attention mechanisms or token representations prevents the model from directly observing and learning structural information like GNNs, thus restricting further improvement on performance.

To address the issues present in LLM-based graph learners and bridge the gap of natural language based graph learning, we propose **InstructGLM** (**Instruct**ion-finetuned **G**raph **L**anguage **M**odel). Given that LLMs have been dominant in many AI domains, we aim to answer the question: **Can LLMs also replace GNNs as the foundation model in graph machine learning?** Intuitively, as one of the most expressive medium, natural language is adept at describing complex structures such that InstructGLM owns following advantages over GNNs:

1) *Flexibility*. A natural language sentence is capable of effectively describing the connectivity at any desired hop levels and intermediate paths without iterative message passing and aggregation. Even multimodal features of the nodes and edges can be directly integrated into natural language prompts, making natural language a very flexible medium to convey both structural and content information on the graph.

2) *Scalability*. Injecting graph structure into multiple natural language sentences enables mini-batch training and independent gradient propagation, which further allows easy scalability

to distributed training and inference on massive graphs with low machine communication overhead.

3) *Compatibility*. Aided by structure descriptions, InstructGLM can consistently reformulate various graph learning pipelines as language modeling tasks, thus fits well into the LLM-based multimodal processing framework, paving the way to integrate graph learning with other AI tasks such as vision, language and recommendation to construct unified AI systems.

In this paper, we focus on tackling node classification, while augmenting it with self-supervised link prediction to enhance the performance. We design a series of scalable graph prompts for generative LLMs (Wei et al., 2021; Chung et al., 2022). Specifically, we systematically employ natural language to describe the graph's topology according to the prompts. The graph structure is clearly and intuitively provided to LLMs without complex pipelines tailored to graphs. Therefore, we can handle graph tasks efficiently and succinctly by the vanilla Transformer architecture (Vaswani et al., 2017) and language modeling objective (Zhang and Sabuncu, 2018) in a generative manner. Overall, our contributions can be summarized by the following four points:

- To the best of our knowledge, we are the first propose to purely using natural language for graph structure representation and perform instruction tuning on a generative LLM to solve graph-related problems. We eliminate the requirement of designing specific complex attention mechanisms tailored for graphs. Instead, we offer a concise and efficient natural language processing interface for graph machine learning, which exhibits high scalability to a unified multimodal and multitask framework, aligning with the current trend in other AI domains.

- Inspired by various message passing mechanisms in GNNs, we have designed a series of rule-based, highly scalable instruction prompts for general graph structure representation and graph machine learning. Although in this paper, our focus lies in exploring instruction tuning on large language models, these prompts can also be used for zero-shot experiments on LLMs.

- We conduct self-supervised link prediction as an generic auxiliary task and further investigate its influence on the primary task under a multitask instruction tuning framework. This exploration holds valuable insights for future LLM-based multitask graph learning, demonstrating the significance of self-supervised link prediction for large language models' better structure understanding on graphs.

- We implement extensive experiments on three widely used datasets: ogbn-arxiv, Cora, and PubMed. The results demonstrate our InstructGLM outperforms previous competitive GNN baselines and Transformer-based methods across all three datasets, achieving the top-ranked performance. These findings validate the effectiveness of our method and underscore the trend of leveraging generative large language models as the foundation model for graph machine learning.

2 Related Work

2.1 GNN-based Methods

Graph Neural Networks (GNNs) (Zhou et al., 2020; Wu et al., 2020; Han et al., 2023a; Wu and Wang, 2022) have been dominant in graph machine learning for a long period. Leveraging message passing and aggregation, GNNs excel in simultaneously learning node features and graph topology. Overall, GNNs with various message passing mechanisms can be categorized as spatial-based ones (Hamilton et al., 2017; Veličković et al., 2017; Xu et al., 2018a; Monti et al., 2017) and spectral-based ones (Kipf and Welling, 2016; Defferrard et al., 2016; Yadati et al., 2019). Inherently, GNNs easily suffer from over-smoothing (Cai and Wang, 2020), with various regularization techniques like MixHop, Jump Knowledge and EdgeDrop (Xu et al., 2018b; Abu-El-Haija et al., 2019; Rong et al., 2019) proposed to mitigate such an overfitting. Another major drawback of GNNs is their inability to directly process non-numeric raw data like text or images, requiring additional feature engineering techniques like BoW, TF-IDF, or Skip-gram as a preprocessing step (Wang et al., 2021). Its lack of compatibility with existing large-scale generative models presents a significant challenge for integration with other AI domains such as vision and language into a unified intelligent system.

2.2 Transformers-based Methods

Attention-based Transformer models can be utilized for graph processing by representing nodes and edges as distinct tokens (Müller et al., 2023).

However, it is computationally intensive for handling large-scale graphs and the global weighted average of attention mechanism can not effectively capture the graph's topology (Kim et al., 2022). To mitigate the issue, some methods incorporate graph structure information into attention matrices (Ying et al., 2021; Park et al., 2022), while others restrict attention to local subgraphs (Nguyen et al., 2022) or ingeniously design graph orthogonal vectors for node and edge tokens(Kim et al., 2022). These newly designed complex pipelines result in indirect representation of graph structure and significantly increasing the learning difficulty. The only work similar to ours is Zhang et al. (2021a), which utilizes natural language templates tailored to biological concept linking (Sokal and Crovello, 1970; Wang et al., 2023b). However, it is difficult for extension beyond classification due to the use of encoder-only model (Liu et al., 2019). Additionally, its natural language templates are not designed for general graph learning thus not as expressive and flexible as ours.

2.3 Fuse GNN and Transformers

GNNs excel at learning structure, while Transformers are proficient in capturing multi-modality features. To combine the advantages of both, Chien et al. (2021) and Duan et al. (2023) utilizes multi-neighbor prediction and LoRa (Hu et al., 2021), respectively, to incorporate graph structure into language models, generating enhanced feature for downstream GNNs. Mavromatis et al. (2023) employs GNNs to perform knowledge distillation on LMs, Zhao et al. (2023) trains GNNs and LMs iteratively in a variational inference framework, while Rong et al. (2020) attempts to replace attention heads with GNNs to better capture global information. The main drawback of the aforementioned methods is the lack of decoupling between Transformers and GNNs, results in training multiple models and incurs significant computational overhead (Nguyen et al., 2022). Moreover, the model performance is still susceptible to inherent issues of GNNs, such as over-smoothing (Yang et al., 2020) and the pipeline of multi-model training is usually very complex compared to the simplicity of a single generative LLM framework.

2.4 Large Language Model based Methods

Inspired by the remarkable zero-shot capabilities, leveraging LLMs in graph problems has attracted considerable attention. Existing works have included utilizing LLM to select the most suitable graph processor based on the query (Zhang, 2023), employing LLM's zero-shot explanations for data augmentation to obtain advanced graph features (He et al., 2023), generating prompts and benchmarks for graph construction, evaluation, biology and structural reasoning (Han et al., 2023b; Jiang et al., 2023; Qian et al., 2023; Guo et al., 2023). There are three works sharing similarities with ours. Guo et al. (2023) attempts to complete graph tasks by describing graphs. However, it uses complex formal languages like (Brandes et al., 2013; Himsolt, 1997) but not flexible natural language. Wang et al. (2023a) and Chen et al. (2023) both explore using natural language with LLM for graph problems, with (Wang et al., 2023a) focusing on mathematical problems on small graphs while (Chen et al., 2023) concentrating on node classification in Text-Attributed Graphs (TAGs) (Hu et al., 2020). In comparison, our natural language instruction prompts exhibit better scalability, applicable to both small and large graphs and not limited to specific graph type. Besides, the three related works only explored the basic capability of LLM for graph tasks in a zero-shot setting. Their performance does not surpass GNN baselines for the most of time with the model freezed, merely demonstrating the potential of LLM as an option for graph tasks. By contrast, we successfully bridge this gap by conducting instruction tuning on generative LLMs with simple prompts, achieving experimental results that surpass all competitive GNN baselines.

3 InstructGLM

In this section, we introduce our proposed **InstructGLM**, a framework utilizing natural language for both graph structure and node features description to a generative LLM and further addresses graph-related problems by instruction-tuning. We start with notation setup, followed by an introduction to the instruction prompts' design principles, and then we illustrate the pipeline with further details.

3.1 Preliminary

Formally, a general graph can be represented as $\mathcal{G} = (\mathcal{V}, \mathcal{A}, E, \{\mathcal{N}_v\}_{v\in\mathcal{V}}, \{\mathcal{E}_e\}_{e\in E})$, where $\mathcal{V}$ is the set of nodes, $E \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set, $\mathcal{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$ is the adjacent matrix, $\mathcal{N}_v$ is the node feature of $v \in \mathcal{V}$ and $\mathcal{E}_e$ is the edge feature of $e \in E$. It is worth noting that the node feature and edge feature can be various modalities in diverse forms. For example, node feature can be textual information in citation networks or social networks, visual images in photography graphs, user profile in customer systems, and even video or audio signals in movie networks, while edge feature can be product reviews in user-item interaction graph of recommender systems.

3.2 Instruction Prompt Design

In order to comprehensively convey the structural information of a graph and ensure the adaptability of the created instruction prompts to various types of graphs, we have systematically designed a set of graph description prompts centered around an central node. These prompts can be differentiated based on the following three questions: **i)** What is the largest hop level of neighbor information about the central node in the prompt? **ii)** Does the prompt include node features or edge features? **iii)** For prompts with large (≥ 2) hop level neighbors about the central node, does the prompt encompass information about the intermediate nodes or paths along the corresponding connecting route?

Regarding the first question, prompts can be classified into two types: those exclusively contain 1-hop connection information, and those with a maximum of 2-hop or 3-hop connection details. Prior works have shown that utilizing up to 3-hop connectivity is sufficient for excellent performance (Hamilton et al., 2017; Veličković et al., 2017; Kipf and Welling, 2016), while information beyond 3-hop typically owns a minor impact on improvement and might even lead to negative effects (Zhang et al., 2021b; Cai and Wang, 2020). Therefore, the maximum level of neighbor information included in the prompts is up to three. However, benefiting from the flexibility of natural language, our designed prompts can actually accommodate structural information of any hop level. As for the latter two questions, there are two possible scenarios for each question, i.e., if or not to include the node or edge features in the prompt, and if or not to include the connecting route information in the prompt.

We then denote an instruction prompt as $\mathcal{T}(\cdot)$ such that $\mathcal{I} = \mathcal{T}(v, \mathcal{A}, \{\mathcal{N}_v\}_{v\in\mathcal{V}}, \{\mathcal{E}_e\}_{e\in E})$ is the input natural language sentence to LLM and v is the **central node** of this prompt. For instance, the simplest form of a graph description prompt containing at most 2-hops neighbor information is:

$$\begin{aligned}\mathcal{T}(v, \mathcal{A}) = &\{v\} \text{ is connected with} \\ &\{[v_2]_{v_2\in\mathcal{A}_2^v}\} \text{ within two hops.}\end{aligned}$$

while its most detailed form which includes node features, edge features and corresponding intermediate paths should be:

$$\begin{aligned}&\mathcal{T}(v, \mathcal{A}, \{\mathcal{N}_v\}_{v\in\mathcal{V}}, \{\mathcal{E}_e\}_{e\in E}) = \{(v, \mathcal{N}_v)\} \text{ is} \\ &\quad \text{connected with } \{[(v_2, \mathcal{N}_{v_2})]_{v_2\in\mathcal{A}_2^v}\} \\ &\quad \text{within two hops through } \{[(v_1, \mathcal{N}_{v_1})]_{v_1\in\mathcal{A}_1^v}\} \\ &\quad \text{and featured paths } \{[(\mathcal{E}_{(v,v_1)}, \mathcal{E}_{(v_1,v_2)})] \\ &\quad {}_{v_1\in\mathcal{A}_1^v,\, v_2\in\mathcal{A}_1^{v_1}}\}\text{, respectively.}\end{aligned}$$

where $\mathcal{A}_k^v$ represents the list of node v's k-hop neighbor nodes. Essentially, the above prompt should contain all 2-hop paths with node and edge features like $(v, \mathcal{N}_v) \xrightarrow{\mathcal{E}_{(v,v_1)}} (v_1, \mathcal{N}_{v_1}) \xrightarrow{\mathcal{E}_{(v_1,v_2)}} (v_2, \mathcal{N}_{v_2})$ centering at node v. All our instruction prompts are summarized in Appendix F.

3.3 Generative Instruction Tuning for Node Classification

In prompt engineering (Li and Liang, 2021; Lester et al., 2021; Shin et al., 2020) or in-context learning (Dong et al., 2022), pretrained models are usually frozen. Instruction Tuning (Wei et al., 2021; Chung et al., 2022), however, directly conveys the requirements of downstream tasks to pretrained models by fusing the original input data with task-specific instructional prompts under the framework of multi-prompt training. This facilitates remarkably effective fine-tuning, especially when coupled with human feedback (RLHF) (Ouyang et al., 2022). Instruction Tuning has already become an indispensable technique for fine-tuning the most powerful large language models.

In this paper, we introduce InstructGLM as a multi-prompt instruction-tuning framework for graph learning. Specifically, we employ a generative large language model with an encoder-decoder or decoder-only architecture as the backbone, then **fuse** all of our designed instruction prompts, which are spanning at different hop levels with diverse structural information, together as input to LLM, enabling mutual enhancement among the instructions. By exclusively using natural language to depict graph structures, we succinctly present the graph's geometry to the LLM and provide a pure NLP interface for all graph-related tasks, make them solvable through a unified pipeline in generative manner. Worth noting that we concentrate on solving node classification task in this study. We train InstructGLM to strictly generate the category label in natural language, and the prevalent Negative Log-Likelihood (i.e. NLL) Loss in language modeling are selected as our objective function.

Given $\mathcal{G} = (\mathcal{V}, \mathcal{A}, E, \{\mathcal{N}_v\}_{v\in\mathcal{V}}, \{\mathcal{E}_e\}_{e\in E})$ and a specific instruction prompt $\mathcal{T} \in \{\mathcal{T}(\cdot)\}$, we denote $\mathbf{x}$ and $\mathbf{y}$ as LLM's input and target sentence, respectively. Then our pipeline can be formed as:

$$P_\theta\left(\mathbf{y}_j \mid \mathbf{x}, \mathbf{y}_{<j}\right) = \mathrm{LLM}_\theta\left(\mathbf{x}, \mathbf{y}_{<\mathbf{j}}\right),$$

$$\mathbf{x} = \mathrm{Concatenate}(\mathcal{P}; \mathcal{I}; \mathcal{Q})$$

$$\mathcal{L}_\theta = -\sum_{j=1}^{|\mathbf{y}|} \log P_\theta\left(\mathbf{y}_j \mid \mathbf{x}, \mathbf{y}_{<j}\right)$$

where $\mathcal{I} = \mathcal{T}(v, \mathcal{A}, \{\mathcal{N}_v\}_{v\in\mathcal{V}}, \{\mathcal{E}_e\}_{e\in E})$ is the graph structure description centering at node $v \in \mathcal{V}$, $\mathcal{L}$ denotes the NLL loss, $\mathcal{P}$ and $\mathcal{Q}$ are the task-specific instruction prefix and query. Specifically, for node classification, we design $\mathcal{P}$ and $\mathcal{Q}$ for node classification as follows: $\mathcal{P} =$ 'Classify the central node into one of the following categories: [<*All category*>]. Pay attention to the multi-hop link relationships between the nodes.' and $\mathcal{Q} =$ 'Which category should $\{v\}$ be classified as?'. More details of the pipeline are depicted in **Figure 2**.

Our InstructGLM actually shares essential similarities in mechanism with various GNNs, and thus covering their advantages. First, we mix prompts with diverse hop-level information together during training, which is akin to MixHop (Abu-El-Haija et al., 2019) in performing graph convolutions on subgraphs extracted at different hop levels. Second, Jumping Knowledge (Xu et al., 2018b) combines outcomes from different convolution layers via jump connections, which is aligned with our prompts featuring intermediate information and high-hop-level neighbors. Additionally, due to LLM's input length limit, similar to GraphSAGE (Hamilton et al., 2017), we conduct neighbor sampling for the central node when filling the prompts to form a mini-batch training. This operation also resembles graph regularization techniques like DropEdge (Rong et al., 2019) for preventing over-smoothing (Chen et al., 2020a). Furthermore, compared to GNNs, our InstructGLM exhibits stronger expressive capabilities. Even a single graph description that contains intermediate paths and k-hop neighbor information is equivalent to a k-layer GNN in expressiveness. Therefore, InstructGLM can readily accommodate the inductive bias of graph tasks without any alterations on LLM's architecture and pipeline. For instance, since our inputs are centralized graph descriptions that directly exhibit the corresponding multi-hop neighbors, self-attention (Vaswani et al., 2017) applied on such inputs can be seen as an advanced weighted average aggregation mechanism of GATs (Veličković et al., 2017; Li et al., 2021), facilitating InstructGLM to effectively grasp different neighbors' varying importance to the central node.

3.4 Auxiliary Self-Supervised Link Prediction

Both SuperGAT (Kim and Oh, 2022) and DiffPool (Ying et al., 2018) introduce auxiliary link prediction task, thus successfully obtain better node representations and performance for node or graph classification, demonstrating that model's comprehension of graph structure can be significantly enhanced by such an auxiliary task. Inspired by them, also to remove the restriction that our instruction prompts can only treat labeled training nodes as central nodes in single-task semi-supervised learning, we introduce self-supervised link prediction as a foundational auxiliary task for InstructGLM. Given arbitrary hop level and central node, we randomly select a neighbor or non-neighbor at this hop level as the candidate. Then we instruct our model to either discriminate whether there is a connection at this hop level between the central node and the candidate node (discriminative prompt) or directly generate the correct neighbor in a generative manner (generative prompt).

Given $\mathcal{G} = (\mathcal{V}, \mathcal{A}, E, \{\mathcal{N}_v\}_{v\in\mathcal{V}}, \{\mathcal{E}_e\}_{e\in E})$, the pipeline of link prediction aligns exactly with node classification. The only distinction lies in the newly designed task-specific prefix and two different query templates for it. Specifically, we design $\mathcal{P}$ and $\mathcal{Q}$ for link prediction as follows: $\mathcal{P} =$ 'Perform link prediction for the central node. Pay attention to the multi-hop link relationships between the nodes.', $\mathcal{Q}_{generative} =$ 'Which other node will be connected to $\{v\}$ within $\{h\}$ hop?' and $\mathcal{Q}_{discriminative} =$ 'Will $\{\tilde{v}\}$ be connected to $\{v\}$ within $\{h\}$ hop?', where v is the central node, $\tilde{v}$ is the candidate node and h is the specified hop level. We enable arbitrary node to act as central node via self-supervised link prediction and ensure a multi-task multi-prompt framework.

4 Experiments

4.1 Experimental Setup

In this paper, we primarily utilize InstructGLM for node classification, also conduct self-supervised link prediction as an auxiliary task. Specifically, we select the following three popular citation graphs: ogbn-arxiv (Hu et al., 2020), Cora, and PubMed (Yang et al., 2016), in which every node represents an academic paper on a specific topic, with its title and abstract included in raw text format. All of our experiments report test accuracy as our metrics and employ the dataset's default numerical node embedding to extend the LLM's vocabulary by adding node-wise new tokens. Implementation details and elaborated dataset-specific statistics are summarized in Appendix C and D.

4.2 Main Results

Our results achieve single-model state-of-the-art performance, surpassing all single graph learners across all three datasets, including both representative GNN models and graph Transformer models, which demonstrates the promising trend for large language models to serve as the foundation model for graph learning.

4.2.1 ogbn-arxiv

For the ogbn-arxiv, we adopt same dataset splits as in the OGB open benchmark (Hu et al., 2020), i.e. 54%/18%/28%. We select top-ranked GNNs from the OGB Leaderboard[1], including DRGAT, RevGAT and etc., as the baselines (Zhang et al., 2022a; Hamilton et al., 2017; Kipf and Welling, 2016; Li et al., 2020; Han et al., 2023a; Shi et al., 2020; Yu et al., 2022a; Veličković et al., 2017; Sun et al., 2020; Li et al., 2021; Zhang et al., 2023a). Several most powerful Transformer-based single-model graph learners like Graphormer are also considered as compared methods against our InstructGLM. (Kuang et al., 2021; Wu et al., 2023; Ying et al., 2021; Dinh et al., 2022)

We instruction-finetune Flan-T5 (Chung et al., 2022) and Llama-v1 (LoRA) (Touvron et al., 2023; Hu et al., 2021) as the backbone for our InstructGLM. The experimental results in Table 1 demonstrate that both models outperform all the GNNs and Transformer-based methods. Particularly, when using Llama-v1-7b as the backbone on the OGB feature, our InstructGLM attains a **1.54%** improvement over the best GNN method and a **2.08%** improvement over the best Transformer-based method. Meanwhile, we also obtain new **SoTA** performance on the GIANT feature.

Method	OGB	GIANT
MLP	55.50 ± 0.23	73.06 ± 0.11
GAMLP	56.53 ± 0.16	73.35 ± 0.08
GraphSAGE	71.19 ± 0.21	74.35 ± 0.14
GCN	71.74 ± 0.29	73.29 ± 0.01
DeeperGCN	71.92 ± 0.16	–
ALT-OPT	72.76 ± 0.00	–
UniMP	73.11 ± 0.20	–
LEGNN	73.37 ± 0.07	–
GAT	73.66 ± 0.11	74.15 ± 0.05
AGDN	73.75 ± 0.21	76.02 ± 0.16
RvGAT	74.02 ± 0.18	75.90 ± 0.19
DRGAT	74.16 ± 0.07	76.11 ± 0.09
CoarFormer	71.66 ± 0.24	–
SGFormer	72.63 ± 0.13	–
Graphormer	72.81 ± 0.23	–
E2EG	73.62 ± 0.14	–
Flan-T5-base	73.51 ± 0.16	74.45 ± 0.11
Flan-T5-large	74.67 ± 0.08	74.80 ± 0.18
Llama-7b	**75.70 ± 0.12**	**76.42 ± 0.09**

Table 1: Results on ogbn-arxiv. We report accuracy on GNNs (Top), Graph Transformers (Middle) and our InstructGLM with different backbones (Bottom).

4.2.2 Cora & PubMed

Method	Cora	PubMed
MixHop	75.65 ± 1.31	90.04 ± 1.41
GAT	76.70 ± 0.42	83.28 ± 0.12
Geom-GCN	85.27 ± 1.48	90.05 ± 0.14
SGC-v2	85.48 ± 1.48	85.36 ± 0.52
GraphSAGE	86.58 ± 0.26	86.85 ± 0.11
GCN	87.78 ± 0.96	88.90 ± 0.32
BernNet	88.52 ± 0.95	88.48 ± 0.41
FAGCN	88.85 ± 1.36	89.98 ± 0.54
GCNII	88.93 ± 1.37	89.80 ± 0.30
RevGAT	89.11 ± 0.00	88.50 ± 0.05
Snowball-V3	89.59 ± 1.58	91.44 ± 0.59
ACM-GCN+	89.75 ± 1.16	90.96 ± 0.62
Graphormer	80.41 ± 0.30	88.24 ± 1.50
GT	86.42 ± 0.82	88.75 ± 0.16
CoarFormer	88.69 ± 0.82	89.75 ± 0.31
Llama-7b	87.08 ± 0.32	93.84 ± 0.25
Flan-T5-base	**90.77 ± 0.52**	94.45 ± 0.12
Flan-T5-large	88.93 ± 1.06	**94.62 ± 0.13**

Table 2: Results on Cora and PubMed. We report accuracy on GNNs (Top), Graph Transformers (Middle) and our InstructGLM with different backbones (Bottom).

[1] https://ogb.stanford.edu/docs/leader_nodeprop/

Hop Info	Link Prediction	ogbn-arxiv	Cora	PubMed
		Llama-v1-7b	Flan-T5-base	Flan-T5-base
Multi-hop	w/	**75.70%**	**90.77%**	**94.45%**
Multi-hop	w/o	75.37%	87.27%	94.35%
1-hop	w/o	75.25%	86.90%	94.30%
Structure-Free-Tuning	w/o	74.97%	75.65%	94.22%

Table 3: Ablation Study Results. In particular, since Cora is equipped with the sparsest semantic feature (Bag of Words) among the three datasets (ogbn-arxiv with Skip-gram and PubMed with TF-IDF.), we can observe that introducing multi-hop structural information provides the greatest performance gain on Cora.

In terms of the compared methods for Cora and PubMed datasets (He et al., 2023), we select those top-ranked GNNs from the two corresponding benchmarks[2] [3] with 60%/20%/20% train/val/test splits, including Snowball, RevGAT and etc. (Abu-El-Haija et al., 2019; Pei et al., 2020; Wu et al., 2019; He et al., 2021; Bo et al., 2021; Chen et al., 2020b; Luan et al., 2022). Besides, the three most powerful Transformer-based single-model graph learners on these 2 benchmarks, i.e., CoarFormer, Graphormer, and GT (Dwivedi and Bresson, 2020), are also considered.

We instruction-finetune Flan-T5 and Llama-v1 (LoRA) as the backbone for our InstructGLM. The experimental results in Table 2 show that our InstructGLM outperforms all the GNNs and Transformer-based methods. Specifically, InstructGLM achieves a **1.02%** improvement over the best GNN method and a **2.08%** improvement over the best Transformer-based method on Cora dataset, while also achieves a **3.18%** improvement over the best GNN and a **4.87%** improvement over the best Transformer-based method on PubMed dataset.

4.3 Ablation Study

In our experiments, two crucial operations that contributes to the remarkable performance of InstructGLM in node classification are multi-prompt instruction-tuning, which provides multi-hop graph structure information to the LLM, and the utilization of self-supervised link prediction as an auxiliary task. To validate the impact of the two key components on model performance, we conduct ablation experiments on all three datasets, the results are shown in Table 3.

Regarding the *Hop Info* column, *Structure-Free-Tuning* indicates fine-tuning the model on titles and abstracts of the nodes. While *1-hop* and *Multi-hop* mean that we utilize prompts that merely include information from 1-hop neighbors and prompts that include information from neighbors with higher hop levels, respectively. The experimental results show that incorporating multi-hop information and including link prediction task can both enhance the model's performance for node classification.

[2] https://paperswithcode.com/sota/node-classification-on-cora-60-20-20-random

[3] https://paperswithcode.com/sota/node-classification-on-pubmed-60-20-20-random

4.4 Instruction Tuning at Low Label Ratio

In previous experiments, our data splits all ensured a relatively high ratio of labeled training nodes. To further investigate the robustness of our InstructGLM, we conduct experiments on the PubMed dataset using its another widely-used splits with extremely low label ratio. Specifically, we have only 60 training nodes available in this setting thus the label ratio is **0.3%**. Despite the challenge, our InstructGLM successfully achieve new **SoTA** performance with **89.6%** test accuracy on the corresponding leaderboard[4]. Detailed comparison with all competitive baselines and more results analysis under this setting are summarized in Appendix B

5 Conclusions

To the best of our knowledge, this paper is the first one that purely represents graph structure via natural language description then further perform instruction-tuning on generative LLMs to effectively solve graph learning problems, demonstrating the huge potential of LLMs as the foundational model for graphp machine learning. Our InstructGLM outperforms all single-model GNNs and Transformer-based graph learners on ogbn-arxiv, Cora, and PubMed datasets. Overall, InstructGLM provides a powerful natural language processing interface for graph machine learning, with Transformer-based generative LLM and natural language as the driving force, it further contributes to the trend of unifying foundational model architecture and pipeline across multiple areas for the AGI pursuit in the future.

[4] https://paperswithcode.com/sota/node-classification-on-pubmed-with-public

Limitations

The primary limitation of our InstructGLM lies in the input token limit of the large language model (LLM). For example, Flan-T5 can only accept a maximum sentence input length of 512, while Llama allows for 2048. When dealing with large-scale graphs, the instruction prompts we construct may not encompass all high-order neighbors within a single natural language sentence due to the limitations of sentence length. The simplest solution to this problem is to construct multiple graph description sentences for each training node (central node) to enumerate all possible neighbors at corresponding hop level. However, this leads to a rapid increase in the training data volume. In this work, learning from GraphSAGE (Hamilton et al., 2017), we repeatedly perform random sampling from the multi-hop neighbor lists of the central node until the sentence length reaches the input token limit to mitigate this issue. Despite our implementation achieving impressive results, we believe that improved neighbor sampling and selection strategies can help InstructGLM better address graph-related tasks, especially in the context of applications involving extremely large-scale graphs like knowledge graphs (Pan et al., 2023). Also, many valuable works about employing InstructGLM beyond node classification and link prediction are still under exploration, more potential future directions are discussed in Appendix E.

Ethics Statement

Our method is proposed to provide a powerful natural language processing interface for graph machine learning tasks. Under normal and appropriate usage circumstances, there is no obvious evidence or tendency that our method will lead to significant negative societal impacts.

References

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, pages 21–29. PMLR.

Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. 2021. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6836–6846.

Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3950–3957.

Ulrik Brandes, Markus Eiglsperger, Jürgen Lerner, and Christian Pich. 2013. Graph markup language (graphml).

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Chen Cai and Yusu Wang. 2020. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020a. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3438–3445.

I-Fan Chen, Brian King, and Jasha Droppo. 2021. Investigation of training label error impact on rnn-t. *arXiv preprint arXiv:2112.00350*.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020b. Simple and deep graph convolutional networks. In *International conference on machine learning*, pages 1725–1735. PMLR.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. 2023. Exploring the potential of large language models (llms) in learning on graphs. *arXiv preprint arXiv:2307.03393*.

Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S Dhillon. 2021. Node feature extraction by self-supervised multi-scale neighborhood prediction. *arXiv preprint arXiv:2111.00064*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al.

2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Norman Di Palo, Arunkumar Byravan, Leonard Hasenclever, Markus Wulfmeier, Nicolas Heess, and Martin Riedmiller. 2023. Towards a unified agent with foundation models. In *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*.

Tu Anh Dinh, Jeroen den Boef, Joran Cornelisse, and Paul Groth. 2022. E2eg: End-to-end node classification using graph topology and text-based node attributes. *arXiv preprint arXiv:2208.04609*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Keyu Duan, Qian Liu, Tat-Seng Chua, Shuicheng Yan, Wei Tsang Ooi, Qizhe Xie, and Junxian He. 2023. Simteg: A frustratingly simple approach improves textual graph learning. *arXiv preprint arXiv:2308.02565*.

Vijay Prakash Dwivedi and Xavier Bresson. 2020. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. 2020. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33:22092–22103.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 299–315.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

Jiayan Guo, Lun Du, and Hengyu Liu. 2023. Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *arXiv preprint arXiv:2305.15066*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Haoyu Han, Xiaorui Liu, Haitao Mao, MohamadAli Torkamani, Feng Shi, Victor Lee, and Jiliang Tang. 2023a. Alternately optimized graph neural networks. In *International Conference on Machine Learning*, pages 12411–12429. PMLR.

Jiuzhou Han, Nigel Collier, Wray Buntine, and Ehsan Shareghi. 2023b. Pive: Prompting with iterative verification improving graph-based generative capability of llms. *arXiv preprint arXiv:2305.12392*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Mingguo He, Zhewei Wei, Hongteng Xu, et al. 2021. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. *Advances in Neural Information Processing Systems*, 34:14239–14251.

Xiaoxin He, Xavier Bresson, Thomas Laurent, and Bryan Hooi. 2023. Explanations as features: Llm-based features for text-attributed graphs. *arXiv preprint arXiv:2305.19523*.

Michael Himsolt. 1997. Gml: A portable graph file format. Technical report, Technical report, Universitat Passau.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Dongkwan Kim and Alice Oh. 2022. How to find your friendly neighborhood: Graph attention design with self-supervision. *arXiv preprint arXiv:2204.04879*.

Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. 2022. Pure transformers are powerful graph learners. *Advances in Neural Information Processing Systems*, 35:14582–14595.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Weirui Kuang, WANG Zhen, Yaliang Li, Zhewei Wei, and Bolin Ding. 2021. Coarformer: Transformer for large graph via graph coarsening.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. 2021. Training graph neural networks with 1000 layers. In *International conference on machine learning*, pages 6437–6449. PMLR.

Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. 2020. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. 2022. Revisiting heterophily for graph neural networks. *Advances in neural information processing systems*, 35:1362–1375.

Sitao Luan, Mingde Zhao, Xiao-Wen Chang, and Doina Precup. 2019. Break the ceiling: Stronger multi-scale deep graph convolutional networks. *Advances in neural information processing systems*, 32.

Costas Mavromatis, Vassilis N Ioannidis, Shen Wang, Da Zheng, Soji Adeshina, Jun Ma, Han Zhao, Christos Faloutsos, and George Karypis. 2023. Train your own gnn teacher: Graph-aware distillation on textual graphs. *arXiv preprint arXiv:2304.10668*.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124.

Luis Müller, Mikhail Galkin, Christopher Morris, and Ladislav Rampášek. 2023. Attending to graph transformers. *arXiv preprint arXiv:2302.04181*.

Dai Quoc Nguyen, Tu Dinh Nguyen, and Dinh Phung. 2022. Universal graph transformer self-attention networks. In *Companion Proceedings of the Web Conference 2022*, pages 193–196.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2023. Unifying large language models and knowledge graphs: A roadmap. *arXiv preprint arXiv:2306.08302*.

Wonpyo Park, Woonggi Chang, Donggeon Lee, Juntae Kim, and Seung-won Hwang. 2022. Grpe: Relative positional encoding for graph transformer. *arXiv preprint arXiv:2201.12787*.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*.

Chen Qian, Huayi Tang, Zhirui Yang, Hong Liang, and Yong Liu. 2023. Can large language models empower molecular property prediction? *arXiv preprint arXiv:2307.07443*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. 2020. Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems*, 33:12559–12571.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2019. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*.

Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. 2020. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. 2020. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

Shikhar Singh, Nuan Wen, Yu Hou, Pegah Alipoormolabashi, Te-Lin Wu, Xuezhe Ma, and Nanyun Peng. 2021. Com2sense: A commonsense reasoning benchmark with complementary sentences. *arXiv preprint arXiv:2106.00969*.

Robert R Sokal and Theodore J Crovello. 1970. The biological species concept: a critical evaluation. *The American Naturalist*, 104(936):127–153.

Chuxiong Sun, Jie Hu, Hongming Gu, Jinpeng Chen, and Mingchuan Yang. 2020. Adaptive graph diffusion networks. *arXiv preprint arXiv:2012.15024*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023a. Can language models solve graph problems in natural language? *arXiv preprint arXiv:2305.10037*.

Qinyong Wang, Zhenxiang Gao, and Rong Xu. 2023b. Exploring the in-context learning ability of large language model for biomedical concept linking. *arXiv preprint arXiv:2307.01137*.

Yangkun Wang, Jiarui Jin, Weinan Zhang, Yong Yu, Zheng Zhang, and David Wipf. 2021. Bag of tricks for node classification with graph neural networks. *arXiv preprint arXiv:2103.13355*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR.

Nan Wu and Chaofan Wang. 2022. Gtnet: A tree-based deep graph learning architecture. *arXiv preprint arXiv:2204.12802*.

Qitian Wu, Wentao Zhao, Zenan Li, David P Wipf, and Junchi Yan. 2022. Nodeformer: A scalable graph structure learning transformer for node classification. *Advances in Neural Information Processing Systems*, 35:27387–27401.

Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. 2023. Simplifying and empowering transformers for large-graph representations. *arXiv preprint arXiv:2306.10759*.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018a. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018b. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pages 5453–5462. PMLR.

Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. 2019. Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Advances in neural information processing systems*, 32.

Chaoqi Yang, Ruijie Wang, Shuochao Yao, Shengzhong Liu, and Tarek Abdelzaher. 2020. Revisiting over-smoothing in deep gcns. *arXiv preprint arXiv:2003.13663*.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31.

Le Yu, Leilei Sun, Bowen Du, Tongyu Zhu, and Weifeng Lv. 2022a. Label-enhanced graph neural network for semi-supervised node classification. *IEEE Transactions on Knowledge and Data Engineering*.

Lu Yu, Shichao Pei, Lizhong Ding, Jun Zhou, Longfei Li, Chuxu Zhang, and Xiangliang Zhang. 2022b. Sail: Self-augmented graph contrastive learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8927–8935.

Jiawei Zhang. 2023. Graph-toolformer: To empower llms with graph reasoning ability via prompt augmented by chatgpt. *arXiv preprint arXiv:2304.11116*.

Jiayou Zhang, Zhirui Wang, Shizhuo Zhang, Megh Manoj Bhalerao, Yucong Liu, Dawei Zhu, and Sheng Wang. 2021a. Graphprompt: Biomedical entity normalization using graph-based prompt templates. *arXiv preprint arXiv:2112.03002*.

Lei Zhang, Xiaodong Yan, Jianshan He, Ruopeng Li, and Wei Chu. 2023a. Drgcn: Dynamic evolving initial residual for deep graph convolutional networks. *arXiv preprint arXiv:2302.05083*.

Wentao Zhang, Zeang Sheng, Yuezihan Jiang, Yikuan Xia, Jun Gao, Zhi Yang, and Bin Cui. 2021b. Evaluating deep graph neural networks. *arXiv preprint arXiv:2108.00955*.

Wentao Zhang, Ziqi Yin, Zeang Sheng, Yang Li, Wen Ouyang, Xiaosen Li, Yangyu Tao, Zhi Yang, and Bin Cui. 2022a. Graph attention multi-layer perceptron. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4560–4570.

Yiyuan Zhang, Kaixiong Gong, Kaipeng Zhang, Hongsheng Li, Yu Qiao, Wanli Ouyang, and Xiangyu Yue. 2023b. Meta-transformer: A unified framework for multimodal learning. *arXiv preprint arXiv:2307.10802*.

Zaixi Zhang, Qi Liu, Qingyong Hu, and Chee-Kong Lee. 2022b. Hierarchical graph transformer with adaptive node sampling. *Advances in Neural Information Processing Systems*, 35:21171–21183.

Zhilu Zhang and Mert Sabuncu. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, 31.

Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2023. Learning on large-scale text-attributed graphs via variational inference. *ICLR*.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81.

A Detailed Pipeline Illustration

Further detailed pipeline is depicted in Figure 2.

B Instruction Tuning at Low Label Ratio

Method	Accuracy
GraphSAGE	76.8 ± 0.9
GAT	79.0 ± 1.4
Snowball	79.2 ± 0.3
GCN	80.4 ± 0.4
SuperGAT	81.7 ± 0.5
ALT-OPT	82.5 ± 1.7
GRAND	82.7 ± 0.6
SAIL	83.8 ± 0.1
ANS-GT	79.6 ± 1.0
NodeFormer	79.9 ± 1.0
SGFormer	80.3 ± 0.6
Llama-7b	85.1 ± 0.6
Flan-T5-base	<u>88.2 ± 0.3</u>
Flan-T5-large	**89.6 ± 0.4**

Table 4: Results on PubMed with 60 training nodes. We report accuracy on GNNs (Top), Graph Transformers (Middle) and our InstructGLM with different backbones (Bottom).

To further investigate the scalability and robustness of our InstructGLM, we conduct experiments

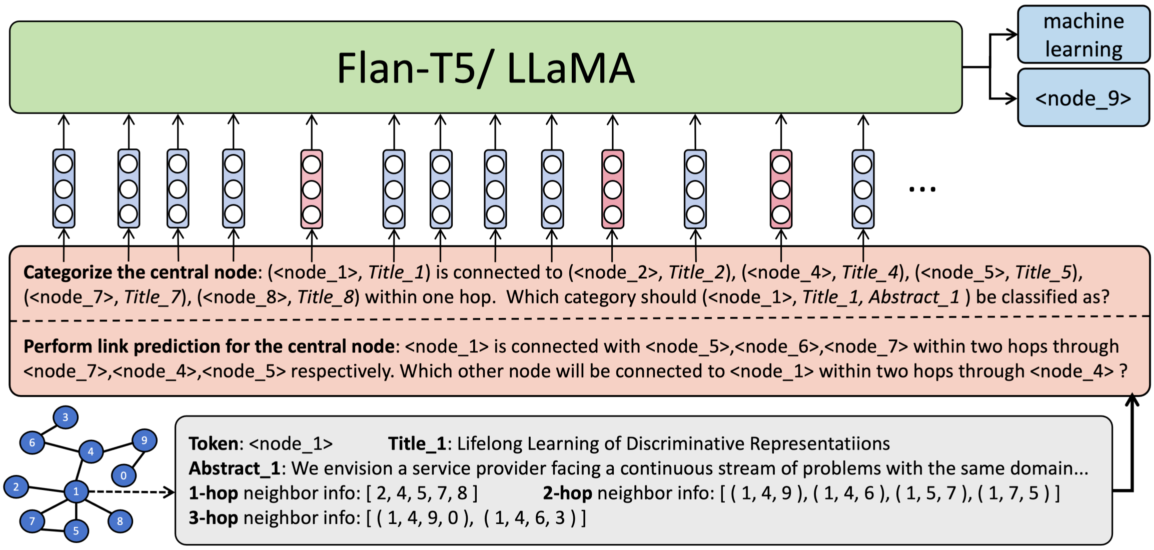

Figure 2: Illustration of InstructGLM. We use graph prompts to describe each node's multi-hop connectivity and meta features in a scalable mini-batch manner, conveying graph structure concisely and intuitively by pure natural language for learning. Subsequently, we instruct LLM to generate responses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

on the PubMed dataset using its another widely-used splits with extremely low label ratio. Specifically, we have only 60 training nodes available in this setting thus the label ratio is **0.3%**.

We consider top-ranked GNNs from the corresponding leaderboard[5], including SAIL, ALT-OPT, GRAND etc., as the GNN baselines. (Luan et al., 2019; Kim and Oh, 2022; Feng et al., 2020; Han et al., 2023a; Yu et al., 2022b) We also include the three most outstanding Transformer-based graph learners under this dataset setting, i.e. ANS-GT, NodeFormer and SGFormer. (Zhang et al., 2022b; Wu et al., 2022, 2023) We then instruction-finetune Flan-T5 and Llama as the backbone for our InstructGLM. The experimental results in Table 4 demonstrate that InstructGLM outperforms all the GNNs methods with an improvement of **5.8%** against the best GNN baseline, while also surpassing the best Transformer-based model by **9.3%**, successfully achieve new **SoTA** performance on the leaderboard.

C Implementation Details

We employ a multi-prompt instruction-tuning framework for all of our experiments and report test accuracy as our metric. Also, we employ a simple MLP over the default feature embedding of the node tokens to align their dimension with the natural language word token embeddings. All of all our experiments are conducted on four 40G A100 GPUs.

For ogbn-arxiv dataset, we adopt the same dataset splits as in the OGB open benchmark (Hu et al., 2020), which is 54%/18%/28%. It takes 3.5 hours per epoch for Flan-T5-Large and 6 hours per epoch for Llama-7b during training. For Cora and PubMed datasets, we use the version that contains raw text information proposed in (He et al., 2023) and employ a 60%/20%/20% train/val/test splits for our experiments. It takes about 1.5 hours per epoch for Flan-T5-Large **(770M)** and 2.5 hours per epoch for Llama-v1-7b-LoRA **(18M)** during training.

To investigate InstructGLM's performance under low-label-ratio training setting, following Yang et al. (2016), we conduct further experiments on the PubMed dataset with the fixed 20 labeled training nodes per class at a 0.3% label ratio, and it takes about 5 minutes per epoch for Flan-T5-Large and 15 minutes per epoch for Llama-v1-7b during training due to limited labeled data.

For both normal setting and low-label-ratio setting, the inference time is about 35ms on Flan-T5-Large and 450ms on Llama-7b per graph prompt sentence.

In terms of hyper-parameter selection, we perform grid search within the specified range for the following parameters: (learning rate: 1e-5, 3e-5, 8e-5, 1e-4, 3e-4, 1e-3), (batch size: 32, 64, 128, 256, 512). We employed the AdamW (Loshchilov and Hutter, 2017) optimizer with a weight decay at 0. All experiments are conducted with 4 epochs.

[5] https://paperswithcode.com/sota/node-classification-on-pubmed-with-public

Dataset	#Node	#Edge	#Class	Default Feature	#Features
ogbn-arxiv	169,343	1,166,243	40	Skip-gram / GIANT	128 / 768
Cora	2,708	5,429	7	Bag of Words	1433
PubMed	19,717	44,338	3	TF-IDF	500

Table 5: Dataset Statistics

D Dataset Statistics

The detailed statistics of the datasets are shown in Table 5.

E Detailed Discussions on Future Work

In this paper, we conduct extensive experiments on Text-Attributed Graphs (TAG) to showcase the powerful capabilities of our proposed InstructGLM in solving graph machine learning problems. Our instruction prompts designed to describe graph structures in natural language demonstrate high generality and scalability, making them applicable to almost all types of graphs. Potential valuable future work can be explored along three dimensions:

- For TAGs, our experiments only used the default OGB-feature embeddings. Future work can consider using more advanced TAG-related embedding features such as LLM-based features like TAPE (He et al., 2023) and SimTeG (Duan et al., 2023). Additionally, leveraging LLM for Chain-of-Thought (Wei et al., 2022), structure information summary, and other data augmentation techniques to generate more powerful instruction prompts will be a promising research direction for graph language models.

- InstructGLM can be integrated into frameworks like GAN and GLEM (Goodfellow et al., 2014; Zhao et al., 2023) for multi-model iterative training, or utilize off-the-shelf GNNs for knowledge distillation (Mavromatis et al., 2023). Also, classic graph machine learning techniques like label reuse, Self-Knowledge Distillation (Self-KD), Correct & Smooth can further enhance the model's performance.

- Benefiting from the powerful expressive ability of natural language and the highly scalable design of our instruction prompts, InstructGLM can be easily extended within a unified generative language modeling framework to various kinds of graphs, addressing a wide range of graph learning problems. For instance, our designed instruction prompts can be further used for link prediction and inductive node classification tasks. And only with slight modifications to our prompts, tasks such as graph classification, intermediate node & path prediction and even relation-based question answering tasks in knowledge graphs with rich edge features are potentially to be effectively deployed.

F Instruction Prompts

In this appendix, we present all our designed instruction prompts. It is worth noting that we follow the following conventions when numbering the prompts:

- The length of each prompt number is 4.

- The first digit represents the task index, where 1 represents the node classification task and 2 represents the link prediction task.

- The second digit represents whether node features or edge features (such as text information) other than numerical feature embedding are used in the prompt. 1 means not used and 2 means used.

- The third digit represents the maximum hop order corresponding to the structural information considered in this prompt. 1 represents only the 1-hop neighbors are included, while 2 and 3 represent the structural information including 2-hop and 3-hop neighbors, respectively.

- The fourth digit represents whether the intermediate node information (i.e. the path) in the high-order connection is considered in this prompt. If the digit is even, it means that the intermediate node is considered, while an odd digit indicates otherwise.

- Specially, in node classification task, we designed a graph-structure-free prompt and numbered it as 1-0-0-0.

1310 ## F.1 Node Classification

1311 **Task-specific prefix:**

Classify the paper according to its topic into one of the following categories:{{All Category List}}.\n Node represents academic paper with a specific topic, link represents a citation between the two papers. Pay attention to the multi-hop link relationship between the nodes.

1312 **Prompt ID: 1-1-1-1**

Input template:

{{central node}} is connected with {{1-hop neighbor list}} within one hop. Which category should {{central node}} be classified as?

Target template: {{category}}

1313 **Prompt ID: 1-1-2-1**

Input template:

{{central node}} is connected with {{2-hop neighbor list}} within two hops. Which category should {{central node}} be classified as?

Target template: {{category}}

1314 **Prompt ID: 1-1-2-2**

Input template:

{{central node}} is connected with {{2-hop neighbor list}} within two hops through {{the corresponding 1-hop intermediate node list}}, respectively. Which category should {{central node}} be classified as?

Target template: {{category}}

1315 **Prompt ID: 1-1-3-1**

Input template:

{{central node}} is connected with {{3-hop neighbor list}} within three hops. Which category should {{central node}} be classified as?

Target template: {{category}}

1316 **Prompt ID: 1-1-3-2**

Input template:

{{central node}} is connected with {{3-hop neighbor list}} within three hops through {{the corresponding 2-hop intermediate path list}}, respectively. Which category should {{central node}} be classified as?

Target template: {{category}}

1317 **Prompt ID: 1-2-1-1**

Input template:

({{central node}},{{text feature}}) is connected with {{1-hop neighbor list attached with text feature}} within one hop. Which category should ({{central node}},{{text feature}}) be classified as?

Target template: {{category}}

Prompt ID: 1-2-2-1 1318

Input template:

({{central node}},{{text feature}}) is connected with {{2-hop neighbor list attached with text feature}} within two hops. Which category should ({{central node}},{{text feature}}) be classified as?

Target template: {{category}}

Prompt ID: 1-2-2-2 1319

Input template:

({{central node}},{{text feature}}) is connected with {{2-hop neighbor list attached with text feature}} within two hops through {{the corresponding 1-hop intermediate node list attached with text feature}}, respectively. Which category should ({{central node}},{{text feature}}) be classified as?

Target template: {{category}}

Prompt ID: 1-2-3-1 1320

Input template:

({{central node}},{{text feature}}) is connected with {{3-hop neighbor list attached with text feature}} within three hops. Which category should ({{central node}},{{text feature}}) be classified as?

Target template: {{category}}

Prompt ID: 1-2-3-2 1321

Input template:

({{central node}},{{text feature}}) is connected with {{3-hop neighbor list attached with text feature}} within three hops through {{the corresponding 2-hop intermediate path list attached with text feature}}, respectively. Which category should ({{central node}},{{text feature}}) be classified as?

Target template: {{category}}

Prompt ID: 1-0-0-0 1322

Input template:

{{central node}} is featured with its {{text feature}}. Which category should {{central node}} be classified as?

Target template: {{category}}

1323 ## F.2 Link Prediction

1324 ### Task-specific prefix:

Perform Link Prediction for the central node:\n Node represents academic paper with a specific topic, link represents a citation between the two papers. Pay attention to the multi-hop link relationship between the nodes.

1325 ### Prompt ID: 2-1-1-1

Input template:

{{central node}} is connected with {{1-hop neighbor list}} within one hop. Will {{candidate node}} be connected with {{central node}} within one hop?

Target template: {{yes/no}}

1326 ### Prompt ID: 2-1-1-2

Input template:

{{central node}} is connected with {{1-hop neighbor list}} within one hop. Which other node will be connected to {{central node}} within one hop?

Target template: {{node_id}}

1327 ### Prompt ID: 2-1-2-1

Input template:

{{central node}} is connected with {{2-hop neighbor list}} within two hops. Will {{candidate node}} be connected to {{central node}} within two hops?

Target template: {{yes/no}}

1328 ### Prompt ID: 2-1-2-2

Input template:

{{central node}} is connected with {{2-hop neighbor list}} within two hops through {{the corresponding 1-hop intermediate node list}}, respectively. Will {{candidate node}} be connected to {{central node}} within two hops through {{the specified 1-hop intermediate node}}?

Target template: {{yes/no}}

1329 ### Prompt ID: 2-1-2-3

Input template:

{{central node}} is connected with {{2-hop neighbor list}} within two hops. Which other node will be connected to {{central node}} within two hops?

Target template: {{node_id}}

1330 ### Prompt ID: 2-1-2-4

Input template:

{{central node}} is connected with {{2-hop neighbor list}} within two hops through {{the corresponding 1-hop intermediate node list}}, respectively. Which other node will be connected to {{central node}} within two hops through {{the specified 1-hop intermediate node}}?

Target template: {{node_id}}

Prompt ID: 2-1-3-1 1331

Input template:

{{central node}} is connected with {{3-hop neighbor list}} within three hops. Will {{candidate node}} be connected with {{central node}} within three hops?

Target template: {{yes/no}}

Prompt ID: 2-1-3-2 1332

Input template:

{{central node}} is connected with {{3-hop neighbor list}} within three hops through {{the corresponding 2-hop intermediate path list}}, respectively. Will {{candidate node}} be connected to {{central node}} within three hops through {{the specified 2-hop intermediate path}}?

Target template: {{yes/no}}

Prompt ID: 2-1-3-3 1333

Input template:

{{central node}} is connected with {{3-hop neighbor list}} within three hops. Which other node will be connected to {{central node}} within three hops?

Target template: {{node_id}}

Prompt ID: 2-1-3-4 1334

Input template:

{{central node}} is connected with {{3-hop neighbor list}} within three hops through {{the corresponding 2-hop intermediate path list}}, respectively. Which other node will be connected to {{central node}} within three hops through {{the specified 2-hop intermediate path}}?

Target template: {{node_id}}

Prompt ID: 2-2-1-1 1335

Input template:

({{central node}},{{text feature}}) is connected with {{1-hop neighbor list attached with text feature}} within one hop. Will ({{candidate node}},{{candidate text feature}}) be connected to ({{central node}},{{text feature}}) within one hop?

Target template: {{yes/no}}

1336 **Prompt ID: 2-2-1-2**

Input template:

({{central node}},{{text feature}}) is connected with {{1-hop neighbor list attached with text feature}} within one hop. Which other node will be connected to ({{central node}},{{text feature}}) within one hop?

Target template: {{node_id}}

1337 **Prompt ID: 2-2-2-1**

Input template:

({{central node}},{{text feature}}) is connected with {{2-hop neighbor list attached with text feature}} within two hops. Will ({{candidate node}},{{candidate text feature}}) be connected to ({{central node}},{{text feature}}) within two hops?

Target template: {{yes/no}}

1338 **Prompt ID: 2-2-2-2**

Input template:

({{central node}},{{text feature}}) is connected with {{2-hop neighbor list attached with text feature}} within two hops through {{the corresponding 1-hop intermediate node list attached with text feature}}, respectively. Will ({{candidate node}},{{candidate text feature}}) be connected to ({{central node}},{{text feature}}) within two hops through ({{the specified 1-hop intermediate node attached with text feature}})?

Target template: {{yes/no}}

1339 **Prompt ID: 2-2-2-3**

Input template:

({{central node}},{{text feature}}) is connected with {{2-hop neighbor list attached with text feature}} within two hops. Which other node will be connected to ({{central node}},{{text feature}}) within two hops?

Target template: {{node_id}}

1340 **Prompt ID: 2-2-2-4**

Input template:

({{central node}},{{text feature}}) is connected with {{2-hop neighbor list attached with text feature}} within two hops through {{the corresponding 1-hop intermediate node list attached with text feature}}, respectively. Which other node will be connected to ({{central node}},{{text feature}}) within two hops through ({{the specified 1-hop intermediate node attached with text feature}})?

Target template: {{node_id}}

Prompt ID: 2-2-3-1 1341

Input template:

({{central node}},{{text feature}}) is connected with {{3-hop neighbor list attached with text feature}} within three hops. Will ({{candidate node}},{{candidate text feature}}) be connected with ({{central node}},{{text feature}}) within three hops?

Target template: {{yes/no}}

Prompt ID: 2-2-3-2 1342

Input template:

({{central node}},{{text feature}}) is connected with {{3-hop neighbor list attached with text feature}} within three hops through {{the corresponding 2-hop intermediate path list attached with text feature}}, respectively. Will ({{candidate node}},{{candidate text feature}}) be connected to ({{central node}},{{text feature}}) within three hops through {{the specified 2-hop intermediate path attached with text feature}}?

Target template: {{yes/no}}

Prompt ID: 2-2-3-3 1343

Input template:

({{central node}},{{text feature}}) is connected with {{3-hop neighbor list attached with text feature}} within three hops. Which other node will be connected to ({{central node}},{{text feature}}) within three hops?

Target template: {{node_id}}

Prompt ID: 2-2-3-4 1344

Input template:

({{central node}},{{text feature}}) is connected with {{3-hop neighbor list attached with text feature}} within three hops through {{the corresponding 2-hop intermediate path list attached with text feature}}, respectively. Which other node will be connected to ({{central node}},{{text feature}}) within three hops through {{the specified 2-hop intermediate path attached with text feature}}?

Target template: {{node_id}}