GOFA: A Generative One-For-All Model for Joint Graph Language Modeling

Lecheng Kong[1*]   **Jiarui Feng**[1*]   **Hao Liu**[1*]   **Chengsong Huang**[1]   **Jiaxin Huang**[1]
Yixin Chen[1]   **Muhan Zhang**[2†]

{jerry.kong, feng.jiarui, liuhao, chengsong, jiaxinh, ychen25}@wustl.edu
muhan@pku.edu.cn
[1]Washington University in St. Louis   [2]Peking University

Abstract

Foundation models, such as Large Language Models (LLMs) or Large Vision Models (LVMs), have emerged as one of the most powerful tools in the respective fields. However, unlike text and image data, graph data do not have a definitive structure, posing great challenges to developing a Graph Foundation Model (GFM). For example, current attempts at designing general graph models either transform graph data into a language format for LLM-based prediction or still train a GNN model with LLM as an assistant. The former can handle unlimited tasks, while the latter captures graph structure much better—yet, no existing work can achieve both simultaneously. In this paper, we identify three key desirable properties of a GFM: self-supervised pretraining, fluidity in tasks, and graph awareness. To account for these properties, we extend the conventional language modeling to the graph domain and propose a novel generative graph language model GOFA to solve the problem. The model interleaves randomly initialized GNN layers into a frozen pre-trained LLM so that the semantic and structural modeling abilities are organically combined. GOFA is pre-trained on newly proposed graph-level next-word prediction, question-answering, and structural tasks to obtain the above GFM properties. The pre-trained model is further fine-tuned on downstream tasks to obtain task-solving ability. The fine-tuned model is evaluated on various downstream tasks, demonstrating a strong ability to solve structural and contextual problems in zero-shot scenarios. The code is available at `https://github.com/JiaruiFeng/GOFA`.

1   Introduction

With the emergence of Large Language Models (LLMs), the field of artificial intelligence is undergoing a profound transformation, shifting from specialized, fragmented models to universal foundation models. A foundation model is pre-trained on large-scale datasets and can be further adapted to diverse downstream tasks using fine-tuning [22] or in-context learning [4, 48]. Foundation models have been developed in different domains to handle text [5, 48], image [30, 3], and even multi-modal data [61, 33, 2]. Because of their versatility and generalizability, these foundation models have become prevalent in these domains.

However, despite preliminary efforts, a foundation model in the *graph domain* has yet to be proposed. In the graph domain, data are highly flexible and dynamic. For example, social networks receive millions of new connections daily [17], and novel molecules and protein structures are frequently

---

[*]Contributed equally. Listing order is random.

[†]Corresponding author

Figure 1: Examples of our pre-training tasks.

discovered [1, 15]. While past researchers have proposed specialized models to learn graph data [57, 29], the models require retraining to accommodate new graphs [8, 39]. Moreover, trained models are usually tied to specific applications and cannot be generalized to new domains and tasks. It becomes increasingly difficult for models to adjust to the ever-evolving nature of graph data. Hence, a graph foundation model that requires minimal data to adapt to new domains/tasks is urgently needed, which has spurred recent endeavors to study general graph models.

The success of LLM inspired a series of preliminary attempts to use LLM to develop general graph models. They can be roughly divided into two categories: LLM as a predictor and LLM as an enhancer [7]. The **LLM as a predictor** approach transforms graph data into representations that LLM can understand and use LLM to generate predictions [46, 6]. However, as suggested by a recent study [51], *such an approach falls short of understanding graph structures*. This inspired the **LLM as an enhancer** approach which adopts LLM to process and unify diverse graph data and feeds them to a GNN to train general graph models [35, 25]. Nevertheless, because GNN outputs fixed-sized representations, they can only handle specific tasks such as classification, and *cannot generalize to a foundational level due to the lack of generation ability*. In summary, the current two approaches cannot fully utilize structural information and be generative simultaneously. We discuss the pros and cons of existing approaches in detail in Section 2.

In this paper, we first identify three desirable properties of a graph foundation model (GFM), namely large-scale self-supervised pretraining, fluidity in tasks, and graph understanding. To achieve the first property, we propose a generic graph self-supervised learning problem similar to the next-token prediction problem in LLMs, allowing label-agnostic and continuous training on any graph data. We then propose a generative model termed Generative One-For-All (**GOFA**) that interleaves GNN layers into an LLM to achieve the second and third properties. Such a design painlessly integrates GNN into an LLM, granting the LLM *graph structural learning ability* while keeping LLM's original free-form text generation ability. Meanwhile, this design allows the pipeline of the original LLM to remain intact, giving the GOFA a *close-to-LLM level of task fluidity*. We pre-train the model with large-scale real-world graph data, Question-Answer (QA) chain data adopted from the NLP domain, and graph structural data to empower the model with the aforementioned foundational abilities in the graph domain (Examples in Figure 1). After pre-training, we further fine-tune the model on downstream tasks. The fine-tuned model is evaluated on various datasets and tasks. Particularly, the GOFA achieved great results on the zero-shot scenario, which demonstrates the strong potential of the GOFA to serve as a graph foundation model.

2 A Desired Foundation Model for Graph

In this section, we summarize three crucial properties a true graph foundation model should possess to motivate our GOFA model design.

Large-scale Self-Supervised Pretraining: One fundamental design of LLM is that it unifies all NLP tasks into a single next-token-prediction paradigm, which enables self-supervised pretraining on large

2

corpus collected from different sources. For pre-training graph models, while numerous efforts have been made from both the LLM as a predictor and LLM as an enhancer approaches, these attempts usually require the learning target to be labeled [35, 7]. However, *a graph foundation model should have no constraint on the input graph (has labels or not) and can learn cross-domain knowledge from large-scale graph data in an unsupervised/self-supervised fashion.*

Fluidity in Tasks: A graph foundation model should also possess the same level of versatility and fluidity in handling different tasks as an LLM. Specifically, such ability can be broken down into three levels: (a) The graph foundation model can naturally respond appropriately to different trained graph tasks according to the user instructions. (b) With appropriate instruction-tuning, the model should have in-context learning ability on unseen tasks. (c) Users can define new tasks unseen in training data by modifying the graph structure and features that fit into the universal input representation of the model. They can continuously train the model on new data without special adaptation. Existing approaches that use GNN models as the predictors are usually either restricted in the output format [35, 53, 18] or need additional fine-tuning on the task head [45, 52]. Consequently, despite having better structural modeling ability, such models cannot accommodate task changes or deal with novel tasks, e.g., shifting from a classification task to a regression task or to a question-answering task that requires outputting all shortest paths between two nodes.

Graph Understanding: Since the LLM as a predictor approach uses a generative LLM to produce output, it naturally has the fluidity to accept varied prompts to tackle different tasks. However, such an approach processes the structural information poorly [51], making the utility of these models minimal on graph. More importantly, even though some recent variants can use auxiliary graph models (such as GNNs) to incorporate structural information [46, 20, 62], the graph models are frozen and not responsive to different prompts, and the output from the graph models may not be the most relevant to the input prompt. On the contrary, *a graph foundation model should account for the unique structural information of graphs such as node degrees, shortest paths, common neighbors, etc., and generate graph representations dependent on the input prompt.* It should not only have LLM's prompt learning capability but also learn graph structure and semantic information jointly.

3 Method

In this section, we first propose a generative modeling framework for graph, serving as the graph counterpart of the traditional language modeling. Next, we introduce a novel GNN-LLM architecture for the proposed graph generative modeling problem. Finally, we describe the unified pre-training and downstream fine-tuning process of the model.

3.1 Generative Modeling for Graph

In plain text applications, aligning the pre-training objective with the input and output format of downstream tasks is straightforward, as they are purely text-based. However, graph data from different domains vary significantly by input feature dimensions and output target. Hence, the first challenge is to *define a unified input and output format for graph tasks*, such that the model can do large-scale self-supervised pre-training on arbitrary graphs and downstream tasks.

For the unified input, following the previous work OFA [35], we extend the definition of Text-Attribute Graph (TAG) beyond graphs that already have text features such as citation and product networks. Specifically, we observe that any node and edge features can be represented by texts. For example, atom and bond descriptions replace the node and edge features in a molecular graph. Furthermore, we can represent non-textual features like pure numbers as texts too, just like they are represented as text strings in LLM. Formally, a TAG is a graph $G = \{V, E, X_V, X_E\}$ where V and E are the sets of nodes and edges. Each node $v \in V$ (edge $e \in E$) corresponds to a text description $x(v) \in X_V$ ($x(e) \in X_E$). Such a format encodes almost all existing graph data and serves well as the universal input representation. For the unified output, we believe that natural language is still the most tangible format for users. Such input and output formats cover almost all commonly seen graphs-related tasks, including classification, regression, and free-form question answering.

Next, we define the generative modeling framework for graphs. Generative modeling for language is to predict the next token given its preceding tokens. However, in graph-based tasks, input sentences are distributed to different nodes and edges in a TAG. To generalize next token prediction to graphs,

we propose to specify Nodes Of Generation (NOG) on graphs as starting points for output generation. Formally, we can define graph generative modeling as the likelihood of the text y associated with the NOG v:

$$p(y|v, G) = \prod_{l=1}^{L} p(y_l|y_{<l}, v, G), \tag{1}$$

where y_l is the l-th token of y, and $y_{<l}$ is its preceding tokens. The NOG v is a learning target with initial corresponding text $x(v)$, and $x(v)$ can be empty. G contains structural and textual information of neighbor nodes to help the model generate y. For example, the sentence completion task is shown on the left of Figure 2, where the text on node v is incomplete, and the goal is to complete the sentence using the existing text and the neighbor information. The question answering task is shown on the right of Figure 2, where a user's question is injected as a node in the graph. The goal is to generate the correct answer. Note that our model is not limited to node-level tasks in conventional graph learning. We discuss more details about data construction in Section 3.3. Users can also specify multiple generation starting points for the same
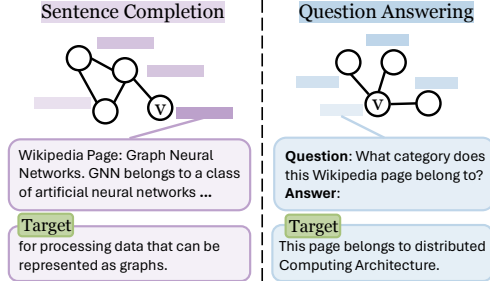

Figure 2: Task examples in TAG. Left: Sentence completion/Next-word prediction. Right: Question Answering. Node v represents NOG.

graph. In addition, when the graph has only one node, the problem degenerates to conventional language modeling.

3.2 GOFA : Generative One-For-All Model

To solve the generative graph modeling problem proposed in Equation 1, we design the GOFA architecture, with an illustration shown in Figure 3. GOFA consists of a graph language encoder and an LLM decoder. In the graph language encoder, we interleave token-level GNN layers with frozen LLM compressor layers. The LLM compressor independently processes the graph's node/edge text features and outputs their compressed fixed-size multi-token embeddings which are fed into a token-level GNN layer. The GNN layer updates the multi-token embeddings by incorporating the input graph structure and feeds the output into the next LLM layer. We alternate GNN and LLM layers to get the final multi-token node representations containing joint structural and semantic information. We then use an LLM decoder to generate texts from the NOG representation. The LLM compressor and decoders are all pre-trained decoder-only transformers. We describe each component in detail as follows.

LLM compressor: Because GNNs require node and edge representations to have the same input dimension, many previous works propose to pool all tokens' output embeddings from the LLM as the node and edge vector representations and feed them to a GNN [35, 25, 20]. While this approach shows effectiveness in tasks of fixed form, such as classification and regression, *it is insufficient in more complex tasks, such as generation tasks*, as 1) the pooling process inevitably loses semantic information, 2) standard LLMs are not trained in a way such that the pooled output embedding is a summarization of the input sentence, and 3) the pooled representation space is no longer compatible with the space of the downstream LLM decoder. Hence, we adopt a sentence compressor [14] that preserves as much information as possible from the original sentence in fixed-size multi-token embeddings. Specifically, the sentence compressor layer has the same architecture as a decoder-only LLM, but the initial sequences of token representations $\{q(x_i)\}_{i=1}^{l}$ are appended by a sequence of K memory tokens $\{q(m_j)\}_{j=1}^{K}$, and the t-th layer of the LLM becomes:

$$\begin{aligned}
\{Q_x^{t+1}, Q_{m,x}^{t+1}\} &= \{q^{t+1}(x_1), ..., q^{t+1}(x_l), q^{t+1}(m_1), ..., q^{t+1}(m_K)\} \\
&= LLM^t(\{q^t(x_1), ..., q^t(x_l), h^t(m_1), ..., h^t(m_K)\}) = LLM^t(\{Q_x^t, H_x^t\}).
\end{aligned} \tag{2}$$

We use Q_x^t and $Q_{m,x}^t$ to represent the t-th LLM layer output corresponding to actual text tokens in x and the K memory tokens appended at the end of text tokens. We use H_x^t to represent the t-th GNN layer output, which will be explained later. In Equation 2, the text tokens (Q_x^t) and memory tokens (H_x^t, processed by the previous GNN layer) are concatenated as a single sequence of embeddings,
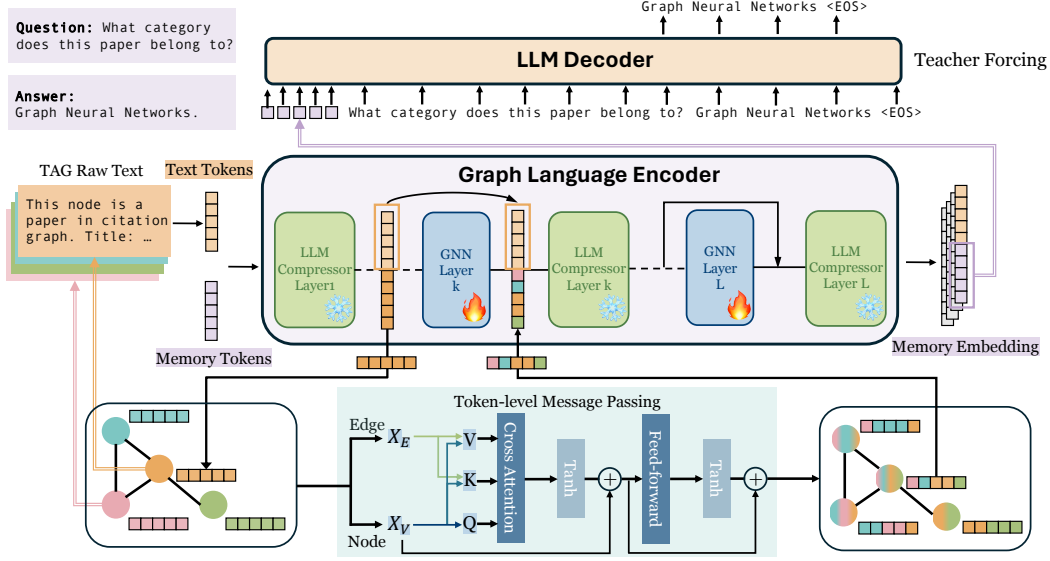
Figure 3: **GOFA Framework**. Text tokens of TAG's node/edges are concatenated with memory tokens to be input to Graph Language Encoder. GNN layers are interleaved into LLM Compressor layers, where memory embeddings from LLM Compressor Layer are used as node/edge features for token-level GNN message passing. Memory embedding will be used for teacher-forcing training.

which are fed to the current LLM layer. *Because the last K tokens attend to all previous tokens, they can compress all information in the sentence into the output embeddings of the K tokens.* This compressor architecture is inspired by ICAE [14]. While the compression capability is not apparent, it is viable through auto-encoder-style fine-tuning, as discussed in Appendix A.1.

Token-level GNN: Conventional GNNs only pass messages between node vector representations, which is incompatible with the vector sequence output $Q_{m,x}$ from our LLM. Hence, we propose a simple extension of conventional GNNs to token level. For node $v \in V$, the t-th GNN layer is

$$H_{x(v)}^t[k] = GNN(Q_{m,x(v)}^t[k], \{(Q_{m,x(u)}^t[k], Q_{m,x(e_{uv})}^t[k])|u \in \mathcal{N}(v)\}), \quad k = 1...K. \quad (3)$$

In the GNN layer, tokens at different indices do not communicate. If we directly stack these GNN layers, they degenerate into multiple isolated GNNs for each token. Nevertheless, we interleave the GNN layers into the LLM layers such that the subsequent self-attention layers on H^t permit information flows between tokens. This approach significantly reduces memory usage because we do not perform cross-token attention in GNN. While edge memory tokens are passed into GNN to assist message passing, their representations are not updated in the GNN layer but directly passed to the next LLM layer, and $H_{x(e)}^t = Q_{m,x(e)}^t$. The specific choice of the GNN can vary. In GOFA , we use a modified Transformer Convolutional GNN [44] to be consistent with the transformer architecture of LLM (see Appendix A.3 for details).

We insert one GNN layer between two transformer layers, while the first and the last layer are always transformer layers. In GOFA , we only insert GNN to the last few transformer layers, but this can be flexible depending on the computational resources. Following previous practice, we incorporate feed-forward (FF) layers into the GNN to increase expressivity and residual connections to stabilize training. Inspired by the gating mechanism in earlier works [21, 2], we apply a $tanh$ gate, initialized at 0, to the GNN and FF layer outputs so that the initial model ignores the information from GNN layers and is equivalent to the pre-trained LLM.

LLM decoder: After applying the model to the textual graph, the memory tokens $Q_{m,x}$ of every node *contain information about the text on the node, the surrounding node text, and the graph structure due to the message-passing process in the GNN layer.* Then, for the NOG v and its corresponding target text y, we insert $Q_{m,x}$ at the front of the token embeddings of the target text to generate and use teacher-forcing to maximize the standard log-likelihood for the next-token-prediction objective.

$$\mathcal{L} = \max_{Q_{m,x(v)}} P(y_1...y_l|Q_{m,x(v)}; \Theta_{\text{DEC}}) = \max_{\Theta_{(\text{GNN,DEC,COMP})}} P(y_1...y_l|v; G; \Theta_{\text{DEC}}, \Theta_{\text{COMP}}, \Theta_{\text{GNN}}). \quad (4)$$

5

In this way, we have modeled the problem in Equation 1. The compressor, decoder, and GNN parameters can be jointly or separately optimized, potentially with parameter-efficient fine-tuning methods like LoRA [22]. In this paper, we use ICAE [14] as our backbone LLM, but the GOFA architecture is not tied to any specific LLM. More details are discussed in Appendix A.2.

3.3 Unified Task Representation in GOFA

The formulation in Equation 1 provides a natural way for users to query the graph by selecting a NOG. However, tasks in the graph domain have different levels, such as node-, link-, and graph-levels, and different tasks have distinct objectives and target nodes. To cover these tasks under the same formulation, inspired by OFA [35], we convert all tasks into subgraph tasks and connect all target nodes in a graph to a virtual prompt node as the NOG for generation (i.e., P nodes in Figure 1). Specifically, for node- and link-level tasks, we sample a k-hop rooted subgraph surrounding the target node or node pairs for node- or link-level tasks and connect the prompt NOG node to target nodes. For graph-level tasks, the prompt NOG node will connect to all nodes in the graph. This design has several advantages: (1) We can specify the prompt node as the NOG input to the LLM decoder for generative modeling. Therefore, the distribution of all tasks can be unified into a single space; (2) The text feature for the prompt node describes the task details. Connecting the prompt node to target nodes helps the prompt node query the most important knowledge from the input graph through attention and message passing. This encourages the model to dynamically learn from the message-passing process subject to prompt instead of only statically learning from the node text $x(v)$.

3.4 Large-Scale Pre-training

In this section, we describe the self-supervised pre-training of GOFA . The goal of the pre-training is to let the GOFA model obtain the ability to query graph structure and context but retain the ability to reason about plain text. Specifically, we perform the pre-training task using MAG240M [23] and Ultrachat200k [9] datasets. Details about the datasets can be found in Appendix C. For each node v in the MAG240M, we sample an ego-subgraph G around v to form a training example. We randomly generate a unique node ID (such as [Node A]) for each node in the training sample and append it to the original node text. This ID will serve as a basis for querying nodes in the graph. We design three pre-training tasks: sentence completion, structural tasks on MAG240M, and question-answering tasks on Ultrachat200k. Figure 1 shows an example of each task. We describe the details of each task below.

Sentence completion task. The objective of the sentence completion task is to train the model to reason about the rest of the text in a node given both the existing text and the information in the rest of the graph. First, we assign the root node as the target for each sampled training graph. Next, we randomly select n nodes in the graph and the target node. All selected nodes' texts are split into halves. The first half forms node text $x(v)$, and the second half becomes the target y to generate. Additionally, we connect a prompt node v_p for each selected node, as shown in Figure 1. Note that for all prompt nodes, we use directed edges to connect them with nodes in the input graph, such that only the information in the input graph will be passed to the prompt node for decoding, and information in different prompt nodes does not influence each other. The text in each prompt node asks the model to complete the sentence of the selected node it connects. This sentence-completion pre-training task adapts LLMs' standard ability to the graph context.

Structural tasks. The objective of the structural tasks is to pre-train GOFA to understand basic graph structural properties. In light of this, we design the shortest paths and common neighbors reasoning tasks between nodes. Specifically, For each training subgraph sample, we randomly sample n nodes as the selected nodes. For each selected node, we ask the model to compute the shortest path distance between it and the root node and output all possible shortest paths between them. Meanwhile, we also ask the model to output the number of common neighbors the selected node and the target node have and the node IDs of their common neighbors. The prompt node v_p will connect to both the target and the selected nodes since our structural tasks need the model to reason about two nodes simultaneously. The text in the prompt node will be the corresponding question. Through these two tasks, the model is expected to gain the ability to identify basic graph structures, which are critical to downstream tasks.

6

Question answering task. Unlike language corpus, which naturally contains many question-and-answer (QA) pairs, graph data usually only contain objective descriptions of entities. Nevertheless, for the model to be fluid in tasks, we need the model to understand user prompts and be sensitive to different tasks. Hence, we synthesize a QA-chain dataset from Ultrachat200k, as shown in Figure 1. A language QA sequence is converted to a chain where nodes with question texts alternate with nodes with answer texts, which are connected by directed edges to represent the conversation order. The last question becomes the text on prompt node v_p, which is connected to every node in the chain, and the last answer is the target text y (see Figure 1 QA-Chain Task for an example). This QA task provides QA pseudo-graphs missing from the common graph corpus, and we found it critical for allowing the model to be responsive to arbitrary tasks expressed in free-form questions.

4 Related work

Prompt Learning in Graph: The success of foundation models inspired many works that aim to adapt their power to the graph domain. Earlier attempts designed a graph prompting mechanism such that a trained model can adapt to new data by fine-tuning a soft prompting vector [36, 58, 45, 53]. GraphPrompt [36, 58] pretrains on link prediction tasks, and then finetune a prompt matrix for each downstream node or graph classification task. All in One [45] designs prompt tokens that are used to modify node features and then take a meta-learning paradigm for multi-task learning. Subsequent works extend graph prompts to allow in-context learning without weight update [25, 13]. For example, Prodigy [25] designs a few-shot prompting graph structure connecting label nodes and data nodes related to support set samples, and trains a GNN for link prediction between label and data nodes. However, these works only tackle limited types of tasks and do not generalize to new domains. Hence, researchers propose integrating LLM into the graph learning and benefit from LLMs' versatility.

LLMs as enhancers: One direction uses LLMs to convert the text features of graphs to unified representations [35, 7, 34, 18] for downstream graph models to distinguish and transfer knowledge between different domains. For example, OFA [35] uses LLM to unify the input features in different datasets and transforms multiple types of graph classification tasks into a unified binary classification format, which allows a single graph model to be trained on diverse tasks and domains. However, OFA could not handle regression tasks. TAPE [18] utilizes LLM to generate question answers and explanations as enhanced node features and use them as input to train a GNN for node classification tasks. Such approaches have good structural modeling ability because of the explicit use of graph structure. However, they usually cannot generate free-form output to handle arbitrary tasks.

LLMs as predictors: Another line of research proposes using LLMs as predictors and aligning graph representation with LLM inputs. Preliminary attempts flatten graphs into text representations and feed them into LLM [7, 66, 16]. These approaches can benefit from LLM for task fluidity but fail to model structural information unique to graph data properly [66, 37, 56]. Alternatively, in molecule modeling, LLM is directly fine-tuned on molecules' SMILE string representations [65, 41]. However, the application for such models is narrow to molecule and chemistry. Realizing this problem, follow-up work extends methods in vision-language domain [2, 33] to the graph domain and train adapters to link graph model outputs to LLM [46, 47, 26, 62, 20]. For example, GraphGPT [46] first implements a text-structure alignment between graph representation and text embedding to pretrain a GNN, then takes GNN output to insert to language space and performs instruction tuning for downstream tasks. Inspired by Q-former [33], GraphTranslator [62] aligns node and text tokens from pre-trained GNN and LLM. UniGraph [20] pretrains GNN using masked word prediction and then tuning a projector to map graph embedding to language space and enable zero-shot learning. However, the GNN and LLM parts of these methods are usually detached, meaning the prompt information can not attend to the message-passing process. More related work about GNNs and transformers is included in Appendix D.

5 Experiment

This section evaluates the proposed methods by answering the following four questions: Q1: Can GOFA use graph information to better model TAG and help complete the target sentence? Q2: Does the pre-trained GOFA help with zero-shot learning? Q3: Which portion of GOFA gives it unique TAG modeling and in-context learning ability? Q4: Can we transfer the prior knowledge in the pre-trained GOFA to downstream fine-tuning tasks?

7

5.1 GOFA pre-training

To answer **Q1**, we pre-train two GOFA models using ICAE models on Llama2-7B [48] and Mistral-7B [28], optimizing the objective in Equation 4. The pre-training datasets include MAG240M [23], UltraChat200k [9], and our proposed structural learning tasks as introduced in Section 3.4. The training details can be found in Appendix F. Figure 4 shows a training loss decrease for the number

of seen tokens. We report the perplexity in Table 1 and compare GOFA with base LLM finetuned on the MAG240M dataset. Note that during pre-training, we only update the weight of the GNN layers, and GOFA 's lower perplexity shows that the structural and semantic information in the node's neighbor can effectively help complete the sentence with more relevance than the original LLM. We observe that fine-tuned LLM does not improve much from the base LLM as it is already well-trained on large-scale academic datasets and cannot directly use the neighbor information to refine the generation. Besides sentence completion, another important GOFA pre-training objective is the structure learning ability; we report shortest path distance and common neighbor count prediction results in Table 1, compared with LLM models whose inputs are textualized graphs, with descriptions of edge connections. We see a significant performance improvement of GOFA over base LLM, showing that a difficult graph task for LLM can be well solved by the GNN layers with better structure modeling ability.

Table 1: Evaluation for pre-trained GOFA . (RMSE for SPD and CN)

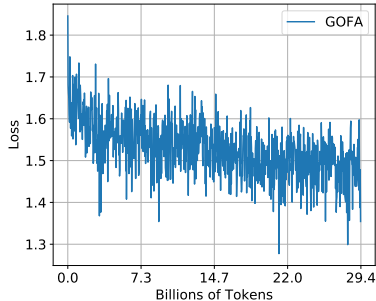|	Perplexity ↓	SPD ↓	CN ↓
Llama2-7B	14.31	1.627	2.162
Mistral-7B	13.03	0.859	1.455
GOFA-Llama	4.844	0.580	0.746
GOFA-Mistral	4.811	0.553	0.531

Figure 4: GOFA training loss.

5.2 Zero-shot learning with GOFA

To answer **Q2**, we performed zero-shot experiments on various graph tasks. Despite using QA-chain data in the pre-training stage, the graph data does not include knowledge about task formats like classification and does not output exact matches to the answers. Hence, we first perform instruction tuning on the pre-trained GOFA in Section 5.1 using two question-template datasets, arxiv, and Pubmed-link. We only use a part of the data, and the goal of instruction fine-tuning is not to let the model learn particular knowledge from these datasets but to make the model understand the task format described in Appendix F.3. We intentionally chose these datasets because the distribution is similar to the pre-training MAG240M dataset, which is important for fair zero-shot experiments.

Table 2: PubMed link instruction tuning on zero-shot setting (Acc).

Task	Cora-Link
Way	2
Llama2-7B	49.72
Mistral-7B	41.34
OFA-Llama2	52.22
GOFA-Llama2-Pubmed	<u>62.36</u>
GOFA-Mistral-Pubmed	**65.91**

Table 3: Zero-shot experiment results with Arxiv instruction tuning (Accuracy). **Bold** and <u>underlined</u> shows best and runner-up results. L2 and M represent Llama-2 and Mistral LLM, respectively.

Task	Cora-Node	PubMed-Node	WikiCS	Products	ExplaGraphs	FB15K237	SceneGraphs
Way	7	3	10	10	2	10	QA
Llama2-7B	29.69	60.95	32.56	50.69	59.02	27.66	38.62
Mistral-7B	54.79	71.02	58.83	61.99	73.03	**63.85**	**45.95**
OFA-Llama2	27.70	56.42	18.5	-	-	-	-
GraphGPT	18.13	70.11	-	-	-	-	-
UniGraph	<u>69.53</u>	<u>72.48</u>	43.45	66.07	-	-	-
GOFA-L2-arxiv-10K	63.56	65.26	47.98	36.85	53.61	28.47	23.52
GOFA-M-arxiv-10K	65.15	64.37	<u>68.19</u>	<u>72.60</u>	<u>78.21</u>	45.81	32.44
GOFA-M-arxiv-40K	**71.20**	**73.11**	**70.49**	**75.83**	**79.56**	<u>55.96</u>	<u>33.06</u>

We perform 10-way classification fine-tuning on only 10k/40k arxiv data for **one epoch** and show zero-shot results in Table 3. While arxiv only contains node classification tasks, we observe that GOFA achieves very non-trivial performance on all node-level (Cora, Pubmed, WikiCS, Products),

link-level (FB15K237), and graph-level (ExplaGraphs, SceneGraphs) tasks. GOFA also generalizes to different ways and even question-answering (SceneGraphs) tasks, showing its desirable fluidity. GOFA outperforms LLM and graph foundation model baselines on most datasets, and exceeds second best results by a large margin ($> 5\%$) on WikiCS, Products, and ExplaGraphs, showing GOFA 's ability to combine the advantage of both LLM and graph models. We also notice that GOFA does not perform as outstanding on the FB15K237 knowledge graph dataset, potentially because it is a link prediction task, which requires task-specific knowledge different from our arxiv instruction tuning data. Using a knowledge graph for tuning can potentially improve the performance and we leave this to future work. We also observe that GOFA is only achieving comparable performance to LLM on the SceneGraph dataset. We notice that SceneGraph contains object coordinate description which makes it easy for LLM to answer tasks about relative location. Whereas, our compressed representation might still discard relevant number information compared to directly prompting this information to LLM. Hence, the advantages of GOFA are not as apparent on tasks where text-only information is sufficient. We discuss potential solutions in Appendix F and leave this to future work.

5.3 Investigation of GOFA architecture

For **Q3**, we observe one unique feature of GOFA is that its GNN layers are embedded into the LLM. A natural question is how important the interleaved GNN layers affect the LLM layer outputs. Hence, we compute a layer-wise Representation Change Ratio value, $\delta = ||H^t - Q^t||/||Q^t||$ (H^t undergoes MLP, residual, and gating modules), quantifying the change that GNN brings to the LLM outputs; a larger value means a higher dependence on graph information. We take the mean ratio of 100 data randomly sampled from each zero-shot dataset. The results in Figure 5 show that almost all GNN layers have a significant impact on the LLM output in both pre-trained and node(arxiv)/link(Pubmed-link) instruction-tuned versions. The pre-trained and arxiv versions have similar importance for each layer as they are all academic

Table 4: Ablation study on prompt edge.

	Single Edge	Double Edge
Cora-Link	49.6	59.4
Cora-Node	58.6	60.2
Products	2.8	22.8
WN18RR	6.4	14.6

Figure 5: Representation changed ratio.

datasets and node-level tasks usually obtain enough information for neighbors within 2-hops. The link-finetuned version has larger third and fourth-layer importance, showing that the link task usually requires the model to consider a larger neighborhood to understand the graph structure.
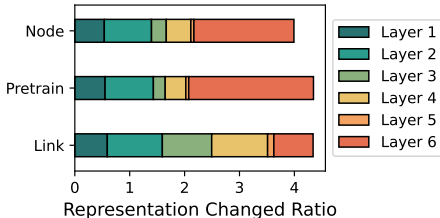
Another important design in GOFA that enables in-context zero-shot learning is the injected prompt node with the user instruction. This allows the GNN to pass messages differently when users provide different prompts. We present a zero-shot ablation study on the connection between target and prompt nodes in Table 4. Single edge means that only the prompt node receives a message from the target nodes, and double edge means that target nodes also receive a message from the prompt node. The former results in static message passing independent of the prompt, and the latter in dynamic message passing. We simultaneously train GOFA -llama on Pubmed-link and arxiv datasets and evaluate their zero-shot performance. Without dynamic message passing, the fine-tuned model can only perform well on one dataset and fail to use its inductive knowledge when the task (Cora-link v.s. Cora-node) or the domain (Cora v.s. Products) shifts. This shows the importance of the prompt node design, which enables dynamic message-passing that can better use the pre-trained knowledge of the GNN and LLM compressor for zero-shot tasks. We answer **Q4** with supervised training in Appendix B.

6 Conclusion, Limitations, and Future works

We introduce GOFA, a generative One-for-All graph foundation model. GOFA is pre-trained in a graph-level next-token prediction manner to enable large-scale self-supervised learning. By integrating GNN layers with LLM layers, GOFA combines the generative capabilities of LLMs for free-form output with the structural learning strengths of GNNs for understanding complex graph connections. Our experiments demonstrate that GOFA, when fine-tuned with a small number of data, achieves impressive zero-shot performance, highlighting its potential as a robust graph foundation model. One limitation of our work is the extensive training time required due to the

use of abundant graph data. Furthermore, interleaving GNN layers with LLM layers increases the embedding dimensions for nodes and edges, which consequently extends the inference time beyond that of a typical LLM. Additionally, we employ a frozen LLM compressor in our architecture; hence, the compression capability is not dynamically integrated with the graph data, potentially impacting the efficiency and adaptability of the model. We believe finetuning a graph language compressor can further enhance the performance of GOFA and will explore it in the future.

References

[1] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, pages 1–3, 2024.

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35: 23716–23736, 2022.

[3] Yutong Bai, Xinyang Geng, Karttikeya Mangalam, Amir Bar, Alan Yuille, Trevor Darrell, Jitendra Malik, and Alexei A Efros. Sequential modeling enables scalable learning for large vision models. *arXiv preprint arXiv:2312.00785*, 2023.

[4] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, S. Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen A. Creel, Jared Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren E. Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas F. Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, O. Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Benjamin Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, J. F. Nyarko, Giray Ogut, Laurel J. Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Robert Reich, Hongyu Ren, Frieda Rong, Yusuf H. Roohani, Camilo Ruiz, Jack Ryan, Christopher R'e, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishna Parasuram Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei A. Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. *ArXiv*, abs/2108.07258, 2021. URL `https://api.semanticscholar.org/CorpusID:237091588`.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`.

[6] Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. Llaga: Large language and graph assistant. *arXiv preprint arXiv:2402.08170*, 2024.

[7] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and Jiliang Tang. Exploring the potential of large language models (llms) in learning on graphs, 2023.

[8] Quanyu Dai, Xiao-Ming Wu, Jiaren Xiao, Xiao Shen, and Dan Wang. Graph transfer learning via adversarial domain adaptation with graph convolution. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4908–4922, 2022.

[9] Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations, 2023.

[10] Jiarui Feng, Yixin Chen, Fuhai Li, Anindya Sarkar, and Muhan Zhang. How powerful are k-hop message passing graph neural networks. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=nN3aVRQsxGd`.

[11] Jiarui Feng, Lecheng Kong, Hao Liu, Dacheng Tao, Fuhai Li, Muhan Zhang, and Yixin Chen. Extending the design space of graph neural networks by rethinking folklore weisfeiler-lehman. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 9029–9064. Curran Associates, Inc., 2023. URL `https://proceedings.neurips.cc/paper_files/paper/2023/file/1cac8326ce3fbe79171db9754211530c-Paper-Conference.pdf`.

[12] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric, 2019.

[13] Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. Towards foundation models for knowledge graph reasoning. In *The Twelfth International Conference on Learning Representations*, 2023.

[14] Tao Ge, Hu Jing, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations*, 2023.

[15] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[16] Jiayan Guo, Lun Du, Hengyu Liu, Mengyu Zhou, Xinyi He, and Shi Han. Gpt4graph: Can large language models understand graph structured data ? an empirical evaluation and benchmarking, 2023.

[17] Stephen J Hardiman and Liran Katzir. Estimating clustering coefficients and size of social networks via random walk. In *Proceedings of the 22nd international conference on World Wide Web*, pages 539–550, 2013.

[18] Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing explanations: LLM-to-LM interpreter for enhanced text-attributed graph representation learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=RXFVcynVe1`.

[19] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering, 2024.

[20] Yufei He and Bryan Hooi. Unigraph: Learning a cross-domain graph foundation model from natural language. *arXiv preprint arXiv:2402.13630*, 2024.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780, nov 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `https://doi.org/10.1162/neco.1997.9.8.1735`.

[22] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=nZeVKeeFYf9`.

[23] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs, 2021.

[24] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*, 2021.

[25] Qian Huang, Hongyu Ren, Peng Chen, Gregor Kržmanc, Daniel Zeng, Percy Liang, and Jure Leskovec. Prodigy: Enabling in-context learning over graphs. *arXiv preprint arXiv:2305.12600*, 2023.

[26] Xuanwen Huang, Kaiqiao Han, Yang Yang, Dezheng Bao, Quanjin Tao, Ziwei Chai, and Qi Zhu. Can gnn be good adapter for llms? *arXiv preprint arXiv:2402.12984*, 2024.

[27] Yinan Huang, Xingang Peng, Jianzhu Ma, and Muhan Zhang. Boosting the cycle counting power of graph neural networks with i^2-GNNs. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=kDSmxOspsXQ`.

[28] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[29] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=SJU4ayYgl`.

[30] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[31] Lecheng Kong, Yixin Chen, and Muhan Zhang. Geodesic graph neural network for efficient graph representation learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=6pC5OtP7eBx`.

[32] Lecheng Kong, Jiarui Feng, Hao Liu, Dacheng Tao, Yixin Chen, and Muhan Zhang. Mag-gnn: Reinforcement learning boosted graph neural network. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 12000–12021. Curran Associates, Inc., 2023. URL `https://proceedings.neurips.cc/paper_files/paper/2023/file/2788b4cdf421e03650868cc4184bfed8-Paper-Conference.pdf`.

[33] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[34] Yuhan Li, Peisong Wang, Zhixun Li, Jeffrey Xu Yu, and Jia Li. Zerog: Investigating cross-dataset zero-shot transferability in graphs. *arXiv preprint arXiv:2402.11235*, 2024.

[35] Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. In *The Twelfth International Conference on Learning Representations*, 2023.

[36] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM Web Conference 2023*, 2023.

[37] Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Michael Galkin, and Jiliang Tang. Graph foundation models. *arXiv preprint arXiv:2402.02216*, 2024.

[38] Péter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020.

[39] Yujie Mo, Liang Peng, Jie Xu, Xiaoshuang Shi, and Xiaofeng Zhu. Simple unsupervised graph representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7797–7805, 2022.

[40] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, pages 4602–4609, 2019.

[41] Chen Qian, Huayi Tang, Zhirui Yang, Hong Liang, and Yong Liu. Can large language models empower molecular property prediction?, 2023.

[42] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.

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

[44] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification, 2021.

[45] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. All in one: Multi-task prompting for graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 2120–2131, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701030. doi: 10.1145/3580305.3599256. URL `https://doi.org/10.1145/3580305.3599256`.

[46] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. Graphgpt: Graph instruction tuning for large language models, 2023.

[47] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Long Xia, Dawei Yin, and Chao Huang. Higpt: Heterogeneous graph language model. *arXiv preprint arXiv:2402.16024*, 2024.

[48] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

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

[50] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`.

[51] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. Can language models solve graph problems in natural language?, 2023.

[52] Song Wang, Kaize Ding, Chuxu Zhang, Chen Chen, and Jundong Li. Task-adaptive few-shot node classification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1910–1919, 2022.

[53] Lianghao Xia, Ben Kao, and Chao Huang. Opengraph: Towards open graph foundation models. *arXiv preprint arXiv:2403.01121*, 2024.

[54] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

[55] Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit S, Guangzhong Sun, and Xing Xie. Graphformers: GNN-nested transformers for representation learning on textual graph. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=yILzFBjR0Y`.

[56] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. Natural language is all a graph needs, 2023.

[57] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=OeWooOxFwDa`.

[58] Xingtong Yu, Zhenghao Liu, Yuan Fang, Zemin Liu, Sihong Chen, and Xinming Zhang. Generalized graph prompt: Toward a unification of pre-training and downstream tasks on graphs, 2023.

[59] Bohang Zhang, Guhao Feng, Yiheng Du, Di He, and Liwei Wang. A complete expressiveness hierarchy for subgraph gnns via subgraph weisfeiler-lehman tests, 2023.

[60] Bohang Zhang, Shengjie Luo, Liwei Wang, and Di He. Rethinking the expressive power of GNNs via graph biconnectivity. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=r9hNv76KoT3`.

[61] Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858*, 2023. URL `https://arxiv.org/abs/2306.02858`.

[62] Mengmei Zhang, Mingwei Sun, Peng Wang, Shen Fan, Yanhu Mo, Xiaoxiao Xu, Hong Liu, Cheng Yang, and Chuan Shi. Graphtranslator: Aligning graph model to large language model for open-ended tasks. *arXiv preprint arXiv:2402.07197*, 2024.

[63] Muhan Zhang and Pan Li. Nested graph neural networks. *Advances in Neural Information Processing Systems*, 34:15734–15747, 2021.

[64] Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Labeling trick: A theory of using graph neural networks for multi-node representation learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=Hcr9mgBG6ds`.

[65] Haiteng Zhao, Shengchao Liu, Chang Ma, Hannan Xu, Jie Fu, Zhi-Hong Deng, Lingpeng Kong, and Qi Liu. Gimlet: A unified graph-text model for instruction-based molecule zero-shot learning, 2023.

[66] Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. Graphtext: Graph reasoning in text space, 2023.

Appendix

A Implementation Details

A.1 In-context autoencoder (ICAE)

This section briefly introduces ICAE and how it helps build the GOFA model; please refer to ICAE paper [14] for the specifics of the model. ICAE contains two decoder-only LLMs. One serves as a language compressor that compresses sentences into a fixed-length sequence of vectors, and the other serves as a language decoder that decodes or queries into the compressed sentence representations. Specifically, during training, an input token sequence $\boldsymbol{x} = \{x_1, ..., x_l\}$ is appended by a K memory tokens $\{m_1, ..., m_k\}$ with trainable embeddings. The concatenated sequence is fed to the LLM compressor with a LoRA adapter [22].

$$\{h(x_1), ..., h(x_l), h(m_1), ..., h(m_K)\} = LLM_{comp}(\{e(x_1), ..., e(x_l), e(m_1), ..., e(m_K)\}), \quad (5)$$

where $e(\cdot)$ and $h(\cdot)$ are the token embeddings and LLM outputs. Then, the decoder LLM only attends to the memory token outputs and tries to decode the original sentence from the memory tokens.

$$\{l(m_1), ..., l(m_K), l(x_1), ..., l(x_l)\} = LLM_{dec}(\{h(m_1), ..., h(m_K), e(x_1), ..., e(x_l)\})$$
$$\min_{\Theta_{comp}} \text{CrossEntropy}(\{l(m_K), l(x_1), ..., l(x_{l-1})\}, \{x_1, ..., x_l\}) \quad (6)$$

The ICAE model is also trained on QA and Language modeling tasks to have more diverse embeddings.

By training this auto-encoder objective on a large-scale, the compressor model learns to compress all information about a sentence to the memory token outputs like in a conventional auto-encoder model. Because the compressed representation contains as much information as possible, GNN can pass messages between nodes with minimal information loss.

A.2 LLM choices of GOFA

Because ICAE preserves as much information in a sentence as possible, we can use it in the GOFA model to comprehensively pass information between sentences, as shown in Section 3.2. However, the GOFA model is not limited to ICAE. Users can first train an ICAE-like objective on any existing LLM and apply the GOFA model to the trained LLM. Or, users can apply the GOFA directly to a pre-trained LLM and train the GOFA without the auto-encoder training. Note that the ICAE architecture has a function similar to an encoder-decoder LLM. We do not use an off-the-shelve encoder-decoder LLM because its encoder output is still subject to the sentence length, which does not fit GNN's need for fixed-sized input.

The design of GOFA can be extended beyond a compressor-decoder architecture. For example, we can have a decoder-only GOFA whose LLM layer is,

$$\{Q_x^{t+1}, Q_{m,x}^{t+1}, Q_y^{t+1}\} = LLM^t(\{Q_x^t, H_x^t, Q_y^t\}), \quad (7)$$

where the GNN is still applied on K memory tokens inserted between the node text x and target text y. This allows the target text to attend to the node text, which may improve the performance of GOFA. However, this formulation forces every node to have a target text, which is usually not what users desire and poses extra computation costs. We will explore this architecture in our future work.

A.3 Transformer Convolutional GNN

As mentioned in Section 3.2, we customize a Transformer Convolutional GNN(TransConv) [44] as the GNN used in Equation 3. Since GNN layers operate on token representations and tokens at different indices do not communicate, we describe the GNN at one index for simplicity. The t-th GNN layer on node i and its neighbors $\mathcal{N}(i)$ is:

$$h^{t+1}(i) = \boldsymbol{W}_o\Big(\sum_{j \in \mathcal{N}(i)} \alpha_{i,j}(\boldsymbol{W}_{v,node}h^t(j) + \boldsymbol{W}_{v,edge}h(e_{i,j}))\Big),$$

$$\alpha_{i,j} = \text{Softmax}\Big(\frac{\boldsymbol{W}_q h^t(i) * (\boldsymbol{W}_{k,node}h^t(j) + \boldsymbol{W}_{k,edge}h(e_{i,j}))}{\sqrt{d}}\Big), \quad (8)$$

$h(\cdot)$ represents input node and edge features. $\boldsymbol{W}$ represents query (q), key (k), value (v), output (o) linear projection for nodes and edges. The formulation closely follows the transformer design [49] and its GNN adaptation [44]. This formulation does not aggregate the last layer embedding $h^t(i)$ into the next layer, because we already add residual to maintain the same effect. We use pre-layer normalization following Llama [48].

B   Supervised Experiment Results

In the supervised experiment, GOFA 's prompt does not include class optional. We show the supervised results in Table 5. GOFA achieved competitive performance on most datasets. In particular, GOFA achieved SOTA performance on the Pubmed dataset, demonstrating that GOFA can transfer pre-trained knowledge to downstream tasks. The fluidity of GOFA also allows it to perform regression tasks, which is not possible for earlier general graph models like OFA [35]. We also notice that GOFA is not performing well on some link datasets, possibly due to the fact that in a supervised setting, we only train a small portion of the data (specific numbers in the experimental details section in Appendix F), and different link tasks require subtle structural information that is difficult to learn and not present in the pre-training tasks. We plan to incorporate more diverse structural tasks into our pre-training datasets for the model to better generalizability.

Table 5: Experiment results in supervised learning.

Task type Metric	Cora Link Acc ↑	Cora Node Acc ↑	PubMed Link Acc ↑	PubMed Node Acc ↑	Arxiv Node Acc ↑	WikiCS Node Acc ↑	WN Link Acc ↑	FB Link Acc ↑	Products Node Acc ↑	ML1M Link RMSE ↓
GCN	78.9±0.6	**82.3**±1.1	77.5±0.4	77.8±0.7	73.9±0.6	77.0±0.6	82.7±0.4	90.1±0.3	80.0±0.7	0.977±0.001
GAT	80.1±0.3	80.4±0.4	80.5±0.2	76.6±0.5	**75.8**±0.3	79.8±0.5	88.8±0.3	93.6±0.1	**81.4**±0.2	**0.923**±0.001
OFA	**87.97**	75.34	**95.89**	77.89	73.44	77.62	**98.31**	95.78	-	-
GOFA-llama	80.96	73.08	79.80	86.54	77.08	78.92	86.62	81.73	75.96	1.197
GOFA-mistral	83.48	77.08	81.65	**87.33**	74.28	**80.34**	88.87	80.11	78.54	1.090

C   Datasets

Cora. The Cora dataset is a co-citation network, where nodes are papers related to artificial intelligence. Edges mean the connected two papers are co-cited by other papers. The Cora dataset contains 2708 nodes and 10556 edges. We collect the Cora dataset and its raw text from OFA [35]. We evaluate the performance of the baseline and our proposed model on Cora for both node-level and link-level tasks. For the node-level task, the aim is to classify the node into the correct paper category from 7 different categories. The split is obtained from OFA. It contains 140/500/2068 samples for train/val/test set respectively. For the link-level task, the object is to predict whether two paper nodes are co-cited or not. We follow the setting of OFA [35] and randomly split all edges into train/val/test sets with a ratio of 0.85/0.05/0.1.

PubMed. The PubMed dataset is a co-citation network, where nodes are papers related to diabetes mellitus. Edges mean the connected two papers are co-cited by other papers. The PubMed dataset contains 19717 nodes and 88648 edges. We collect the PubMed dataset and its raw text from OFA [35]. We evaluate the performance of the baseline and our proposed model on PubMed for both node-level and link-level tasks. For the node-level task, papers have 3 different categories. The goal is to classify the node into the correct paper category. We obtain the split directly from original source. It contains 60/500/19157 samples for train/val/test set respectively. For the link-level task, the object is to predict whether two paper nodes are co-cited or not. We follow the setting of OFA [35] and randomly split all edges into train/val/test sets with a ratio of 0.85/0.05/0.1.

Arxiv. The Arxiv dataset is a citation network, where nodes are papers related to computer science and edges mean two papers have a citation relationship. The Arxiv dataset contains 169343 nodes and 1166243 edges. We collect the Arxiv dataset and its raw text from OGB [24]. We evaluate the node classification on the Arxiv dataset. The goal is to classify the paper node into the correct category from 40 possible categories. We obtain the split directly from OGB [24]. It contains 90941/29799/48603 samples for train/val/test set, respectively.

WikiCS. The WikiCS dataset is a graph obtained from Wikipedia. The nodes in WikiCS are Wikipedia terms and their descriptions. The edges mean there is a hyperlink between two terms. We collect the WikiCS dataset and its raw text from [38]. There are 11701 nodes and 216123 edges in the graph. We evaluate the performance of WikiCS on the node classification task. There are 10 different classes. We follow the same split as OFA [35], which contains 580/1769/5847 samples for the train/val/test set, respectively.

Products. The Products dataset is a co-purchase graph. The nodes in the graph represent product items from the Amazon platform, and the edges represent that two products are co-purchased together. We obtain the Products and their raw texts from TAPE [18], which is a subset from the original ogbn-Products [24] dataset. It contains 54025 nodes and 144638 edges. We evaluate the node classification performance on Products. The data from the original source contains 47 different categories. However, we found that there are two classes that don't have any nodes and one class contains nodes with missing text. Therefore, we exclude these classes. Finally, there are 44 different categories and 14695/1567/36982 samples for the train/val/test set, respectively.

MoiveLens-1M (ML1M). The ML1M dataset is a bipartite graph obtained from a movie rating platform. The nodes in the graph represent a user or a movie. The edges represent the score the user rates to the movie. The rating ranges from 1 to 5. We obtained the ML1M dataset and its raw texts followed PyG [12]. It contains 9923 nodes and 1000209 edges. We evaluate the performance of models on ML1M dataset with a link regression task. Specifically, the task is to predict the score the user give to the movie. For ML1M, we randomly split all edges into train/val/test with a ratio 0.85/0.05/0.1, which results in 850177/50011/100021 samples for the train/val/test set, respectively.

WN18RR. The WN18RR is a knowledge graph generated from WordNet. The nodes in the graph represent English words and edges represent the relationship between two words. We obtain the WN18RR dataset from OFA [35]. The graph contains 40943 nodes and 93003 relations. We evaluate the performance of baselines and our model on the link classification task using WN18RR. There are 11 different relationships in the dataset. We obtain the split directly from OFA [35]. It contains 86835/3034/3134 samples for the train/val/test set, respectively.

FB15K237. The FB15K237 is a knowledge graph generated from Free Base. Nodes in the dataset represent entities in the world and edges represent the relation between entities. We obtained the dataset from OFA [35]. The FB15K237 is used to evaluate the link classification. The dataset contains 237 unique classes. We follow the setting of OFA [35] and split the dataset with a ratio of 0.85/0.05/0.1, which results in a total of 272115/17535/20466 samples for train/val/test set, respectively.

ExplaGraphs. The ExplaGraphs is a graph question answering dataset on commonsense concepts. Nodes in the dataset represent a common sense concept and edges represent the relation between two concepts. We obtain the dataset from G-retriever [19] The ExplaGraphs can be used for question-answering on graphs. We obtain the split directly from G-retriever [19]. It contains 1659/553/554 graph samples from the train/val/test set.

SceneGraphs. The SceneGraphs is a graph question answering dataset on scene graphs. Nodes in the dataset represent an object in an image and edges represent the relation between two objects. We obtain the dataset from G-retriever [19] The SceneGraphs can be used for question-answering on graphs. We obtain the split directly from G-retriever [19]. It contains 59978/19997/20025 graph samples from the train/val/test set.

MAG240M. The MAG240M dataset is a citation network generated from Microsoft Academic Graphs. The nodes represent academic papers and the links represent a citation relation between two papers. We obtained the dataset and raw text from OGB-lsc [23]. However, the original dataset is extremely large and contains nodes without text features (author and institution nodes), since we mainly use the dataset for pre-training, we further downsample the original dataset. Specifically, we only keep paper nodes and citation links between papers. Further, we downsample the edges in the following ways. First, we selected all nodes in the train/val/test split provided by OGB-lsc [23]. Next, we filter the edges through two rounds. In the first round, we only keep the edge if either the source or the target is in the selected nodes. If any node in the added edge is not in the selected nodes, we add it to the node set. Next, in the second round, we include additional edges where both the source and target are in the selected nodes (additional nodes are added in the first round). The procedure results in a total of 5875010 nodes and 26434726 edges.

Ultrachat200k. The Ultrachat200k is a question-answering dataset. each sample is a multi-round conversation obtained from the web. We obtained the Ultrachat200k from [9]. However, the original dataset is not a network. To convert it to a graph dataset, we manually create a graph structure for it. Specifically, if the original sample has k round of conversation, we will generate $k - 1$ graph sample. The i-th graph will contain the first i round of conversation. Each node in the graph is either a question or an answer. The question and answer are linked by a directed edge indicating the order of the conversation. The conversation of $i + 1$ round will be the question-answer pair for this graph. Since we mainly use the dataset for pre-training. We only include *train-sft* subset. After the conversion, there are a total of 449929 graphs in total.

D   Related Work Extended

GNNs and Transformers: In recent years, GNNs have become the most popular method for dealing with graph learning problems due to their extraordinary ability in structural learning. Particularly, Previous works [54, 40] show that the expressive power of message-passing GNNs can be as powerful as the 1-dimensional Weisfeiler-Lehman test, a powerful algorithm for graph isomorphism problems. Many recent works also try to design more powerful GNNs that beyond the 1-WL test [63, 31, 10, 27, 60, 59, 11, 32] for better structural ability like learning distance between nodes or counting cycles in graph. Some works try to combine the GNN with the transformer. particularly, GraphFormers [55] and GROVER [43] also insert a GNN layer between consecutive transformer layers for modeling graph inductive bias. Different from us, their transformer layers are randomly initialized and directly tuned on downstream tasks without text.

E   Graph Structure Question Example of LLM

We assessed the ability of LLMs to respond to questions related to graph structures, including shortest path distances and common neighbor counting. For this evaluation, graph edges were described using plain text, and the LLM was tasked with generating the answers. The results of this evaluation are presented in Table 1. These findings indicate that LLMs struggle to comprehend graph structures effectively. We include examples of the questions posed and the corresponding answers generated by the LLM in Table 6, to illustrate these challenges.

F   Experimental settings

F.1   General settings

Subgraph sampling: In the GOFA, for node/link/graph-level tasks, the input format is unified as a subgraph task. Namely, for node/link-level tasks, we will select a k-hop subgraph surrounding the target nodes as the input graph for the model. We follow a similar subgraph sampling strategy as OFA [35] with minor improvement. Specifically, for node-level tasks, we directly sample the k-hop subgraph rooted at the target node. Meanwhile, we set a threshold for maximum nodes per hop. If the nodes in a certain hop exceed the threshold, we will randomly sample maximum nodes from all nodes. For link-level tasks, we will independently do the sampling for both two nodes and then merge two subgraphs as the final subgraph.

Implementations. Both the GOFA and all baselines are implemented using Python with Pytorch, transformers, and PyG [12] packages.

F.2   Pre-training of GOFA

Dataset. For pretraining, we use two datasets: MAG240M and Ultrachat200k. For MAG240M, each training sample is a subgraph sampled around a node. Next, we will construct sentence completion, Shortest path distance, and common neighbor tasks for each sample. Specifically, given a sampled subgraph rooted at the target node, we will first build the sentence completion task on the target node (the root node in the sampled subgraph). Next, we randomly sample k nodes in the subgraph. Next, for each node, we will independently construct all three tasks. For the shortest path distance and common neighbor, the task is to return the results between the selected nodes and the target node. In the pretraining, we set k as 3, which results in 10 tasks for each training sample. For Ultrachat200k,

Table 6: Comparison of LLM-generated answers with correct answers for graph structure questions.

Question	Answer	Correct Answer
In the graph node 8 connects to 1, node 13 connects to 9, node 8 connects to 7, node 3 connects to 13, node 3 connects to 4, node 6 connects to 6, node 11 connects to 8, node 12 connects to 11, node 2 connects to 12, node 14 connects to 7, node 6 connects to 3, node 6 connects to 9, node 2 connects to 8, node 2 connects to 9, node 5 connects to 8, node 3 connects to 11, node 8 connects to 5, node 10 connects to 11, node 2 connects to 9, node 9 connects to 10, what is a shortest path between node 8 and node 9?	To find the shortest path between node 8 and node 9, we can use Dijkstra's algorithm or any other shortest path algorithm. However, since this graph is small, we can find it manually. Here's one of the shortest paths between node 8 and node 9: Start at node 8. Move from node 8 to node 5 (neighbor). Move from node 5 to node 8 (neighbor). Move from node 8 to node 2 (neighbor). Move from node 2 to node 9 (neighbor). So, the shortest path between node 8 and node 9 is: **8 -> 5 -> 2 -> 9**.	**8 -> 2 -> 9**
In the graph node 8 connects to 1, node 13 connects to 9, node 8 connects to 7, node 3 connects to 13, node 3 connects to 4, node 6 connects to 6, node 11 connects to 8, node 12 connects to 11, node 2 connects to 12, node 14 connects to 7, node 6 connects to 3, node 6 connects to 9, node 2 connects to 8, node 2 connects to 9, node 5 connects to 8, node 3 connects to 11, node 8 connects to 5, node 10 connects to 11, node 2 connects to 9, node 9 connects to 10, what are common neighbors between 13 and 5?	To find the common neighbors between nodes 13 and 5, we need to identify all nodes that are neighbors of both nodes 13 and 5. From the provided connections: Node 13 is connected to node 9 and node 3. Node 5 is connected to node 8. Therefore, **the common neighbor between node 13 and node 5 is node 8**.	**No common neighbors.**

we directly use the constructed graphs for pertaining. The detailed task prompts and answer examples are shown in Table 7.

Training details. The initial weight of the LLM compressor and decoder is obtained from ICAE [14]. The initial weight of all GNN layers is randomly initialized. The value of all gates in the residual connection is set to 0 to ensure the initialized model performs the same as the original language model. During the training, we only tune the GNN layers. Due to limited computation resources, we only include 1500000 training samples from MAG240M and 120000 training samples from Ultrachat200k. We pre-train two models using Llama2-7b and Mistral-7b, respectively. The training is conducted on 8 NVIDIAA100_SXM4_80GB GPUs with DeepSpeed stage 2 [42] parallelism. The detailed training parameters are set the same for both two models and are listed in Table 8. We use AdamW optimizer with $\beta = (0.9, 0.95)$. We use a cosine annealing learning rate scheduler, and the minimum learning rate is 10% of the initial learning rate. We restarted the learning rate 2 times on one-third and two-thirds of the training.

F.3 Zero-shot learning

Setting. For the zero-shot learning, we select Cora-link, Cora-node, PubMed-node, WikiCS, Products, ExplaGraphs, and SceneGraphs as evaluation datasets. For all datasets, we directly evaluate baselines and GOFA on the test set.

Table 7: Detailed question and answer example in pertaining task.

Task	Question example	Answer example
Sentence completion	*Complete the sentence of the target node.* *Complete the sentence of the node[NODE.A].*	*The rest of the sentence in the target node.* *The rest of the sentence in node [NODE.A].*
Shortest paths	*Compute the shortest path distance between the target node [NODE.L] and node [NODE.B] and generate all shortest paths from the target node to the node [NODE.B]. Please separate nodes in the path with ->. If multiple paths exist, generate all of them with an ascending order of node sequences and separate different paths with ;.*	*The shortest path distance is 2. Shortest paths: [NODEID.L] -> [NODEID.G] -> [NODEID.B].*
Common Neighbors	*Is there any common neighbor between the target node [NODE.L] and node [NODE.B]? If it exist, please give the total number and list all common neighbors in ascending order of node, separate nodes with ;.*	*There is 1 common neighbor between two nodes, including [NODEID.G].*
QA-Chain	*What are the rules and restrictions in place for COVID-19 in the city?*	*I don't have any live data regarding the covid-19 rules and restrictions. Please check with the local authorities or health department for the latest guidelines and restrictions in your city.*

Table 8: Hyper-parameters for pretraining.

lr	weight_decay	batch_size	dropout	grad_clip	gradient_accum	llm_max_length	optimizer
0.0001	0.1	8	0.0	0.5	8	128	AdamW

Detail of Baselines. For baseline methods, we compare the GOFA with two types of methods. The first type of method is to evaluate the performance directly using the LLM models. We select Llama2-7B and Mistral-7B [28] as baseline models. For these models, we directly give the well-trained model the content in all target nodes and then concatenate the same prompt we use in the GOFA. The second type of method is the graph foundation model. We select the OFA [35] and GraphGPT [46] as baseline methods. For OFA, we directly run the code from the original source to train the model using the Arxiv and FB15K237 datasets. We train the model 50 epochs and all other settings following the default OFA setting. To ensure a fair comparison, we use the Llama2-7b as the embedding model for OFA and report the test performance. For GraphGPT, we directly report the result from the original paper.

Detail of GOFA. For the GOFA, we fine-tune the model from the pre-training checkpoint. In fine-tuning, we will train the parameters of GNN and LoRA layers in the LLM decoder. To comprehensively evaluate the performance of GOFA, We separately fine-tune the GOFA on different datasets. Specifically, we design two different settings. In the first setting, we fine-tune the model using the Arxiv dataset with the node classification task. In the second setting, we fine-tune the model using the PubMed dataset with the link prediction task. For each setting, we fine-tune both the Llama2-7B and Mistral-7B versions, which results in 4 different versions. We denote as GOFA-Llama2-Node, GOFA-Mistral-Node, GOFA-Llama2-Link, GOFA-Mistral-Link, and GOFA-Mistral-Joint, respectively. For

Table 9: Prompt examples of GOFA for each dataset in Zero-shot learning.

Dataset	Prompt
Cora-node	*You need to answer this question based on the information from this node: [NODEID]. This is a co-citation network focusing on artificial intelligence, nodes represent academic papers and edges represent two papers that are co-cited by other papers. You are an expert in computer science. You need to choose the correct paper category based on the paper content and its co-citation network. For example, if the paper [NODEID] {<label_description>, choose <label>;}. What is the most likely paper category for the target paper? Choose from the following: {<label>}.*
Cora-link	*You need to answer this question based on the information from this node: [NODEID1] and [NODEID2]. This is a co-citation network focusing on artificial intelligence, nodes represent academic papers, and edges represent two papers that are co-cited by other papers. You are a computer science expert tasked with determining whether two given papers are co-cited by another paper based on their content and network characteristics. Evaluate the following criteria: assess whether the topics of the two papers are similar, check if the shortest path distance between the two papers is small, and verify whether the papers have a large number of common neighbors in the citation network. If the answer to most of these questions is Yes, choose Yes; if the answer to most of these questions is No, choose No.*
PubMed-node	*You need to answer this question based on the information from this node: [NODEID]. This is a co-citation network from the Pubmed platform focusing on diabetes mellitus. Nodes represent academic papers and edges represent two papers that are co-cited by other papers. You are an expert on diabetes mellitus. You need to choose the correct paper category based on the paper content and its co-citation network. For example, if the paper [NODEID] {<label_description>, choose <label>;}. What is the most likely paper category for the target paper? Choose from the following: {<label>}.*
WikiCS	*You need to answer this question based on the information from this node: [NODEID]. This is a Wikipedia graph focusing on computer science. Nodes represent Wikipedia terms and edges represent two terms that have hyperlinks. You are an expert in computer science. You need to choose the correct category of Wikipedia term based on the term content. For example, if the term [NODEID] {<label_description>, choose <label>;}. What is the most like category for this Wikipedia entry? Choose from the following: {<label>}.*
Products	*You need to answer this question based on the information from this node: [NODEID]. This is a co-purchase network from the Amazon platform. Nodes represent the products sold on Amazon and edges represent two products that are co-purchased together. For example, if the product [NODEID] {<label_description>, choose <label>;}. What is the most like category for this product? Choose from the following: {<label>}.*
FB15K237	*You need to answer this question based on the information from this node: [NODEID1] and [NODEID2]. This is a knowledge graph from the FreeBase. Nodes represent knowledge entities and edges represent relations between two entities. You are an expert in knowledge graph reasoning. You need to choose the correct relation type between two target entities based on their existing relations. For example, if two relations {<label_description>, choose <label>;}. What is the relationship between two target entities? Choose from the following list: {<label>}."*
ExplaGraphs	*This is a graph constructed from commonsense logic. Nodes represent commonsense objects and edges represent the relation between two objects. You are a logic expert tasked with analyzing the logical relationship between two arguments related to connected entities. Determine if the arguments support or counter each other based on their logical coherence. If there is no logical conflict between the two arguments and they are in agreement, choose Support; if the arguments exhibit a logical conflict or contradiction, choose Counter.*
SceneGraphs	*This is a scene graph generated from an image. Nodes represent an object in the image and edges represent the relationship between two objects. <Question>*

Table 10: Hyper-parameters for zero-shot instruction fine-tuning.

Model	weight_decay	gradient_accum	llm_max_length	train_sample_size
GOFA-Llama2-Node	0.1	4	256	10000
GOFA-Mistral-Node	0.1	64	256	10000/40000
GOFA-Llama2-Link	0.0001	4	256	60000
GOFA-Mistral-Link	0.1	64	256	40000

all evaluation and pre-training datasets, we design multiple prompt templates with instructions to let the model select the correct label from the provided label list. For each label in each dataset, we use the GPT-4 to generate a short description for the label. The detailed prompt examples for all datasets are shown in Table 9. For fine-tuning on Arixv, since it is hard to include all 40 ways in the prompt, we randomly sampled 5 ways during the training for each sample. For each pre-training dataset, we randomly sample a fixed number of training samples in a stratified way. The detailed parameters for fine-tuning are listed in Table 10. All parameters not listed in the table are the same as the pre-training setting. For all training versions, we directly evaluate the model on the test set of all evaluation datasets. We evaluate the model on the whole test set except Products and FB15K237, which only sample 10000 samples for evaluation. For evaluation, we will match the text output generated by the GOFA with the ground true label to compute the accuracy of the classification task. For the regression task, we will extract the number from the output text and compute the metric with the correct value.

F.4 Supervised-learning

Table 11: Hyper-parameters for supervised fine-tuning.

lr	weight_decay	grad_clip	gradient_accum	llm_max_length
0.0001	0.1	0.5	4	256

Setting. For the supervised-learning setting, we select Cora (node/link), PubMed (node/link), Arxiv, WikiCS, WN18RR, FB15K237, Products, and ML1M datasets for the evaluation. For all datasets, we utilize the default split described in Appendix C. To ensure a fair comparison, we employ subgraph sampling for GOFA and all baseline methods. For all datasets, the sampling hop is 3 and the maximum nodes per hop are 5, except for the ML1M, which is 1 and 15 respectively.

Detail of baselines. For the traditional GNN methods, we include GCN [29] and GAT [50]. To ensure a fair comparison, we use Llama2-7B to convert raw texts in all datasets to sentence embedding and use this as the model's input node/edge features. We re-implement both methods in order to adapt the original method with subgraph input. Specifically, for but node/link-level tasks, we will add labeling trick [64] to the target nodes at the beginning. After message passing, we will use the summation pooling on all target nodes and use the result embedding for the prediction. For traditional GNN methods, we train and evaluate each dataset independently. For all datasets, we search the number of layers and dropout parameters. For each parameter set, we repeat the experiment 4 times select the parameter set with the best validation performance, and report the performance on the test set. For the graph foundation model, we include OFA [35] as the baseline. The OFA is simultaneously trained and evaluated on all datasets. To ensure a fair comparison, we get their code from the original source and train the model on Cora (node/link), PubMed (node/link), Arxiv, WikiCS, WN18RR, and FB15K237 dataset using the Llama2-7b as base LLM model. Similarly, for OFA, we use the same subgraph sampling parameters as all other methods. For other parameters, we use the default parameter provided in their code. We only run the model one time and report the final performance.

Detail of GOFA. For the GOFA, we fine-tune the model from the pre-training checkpoint on both Llama2 and Mistral. In fine-tuning, we will train the parameters of GNN and LoRA layers in the LLM decoder. We simultaneously fine-tune the model on the train set of Cora-node, Cora-link, PubMed-node, PubMed-link, Arxiv, WikiCS, WN18RR, FB15K237, Products, and ML1M. For each dataset, we will randomly sample a fixed number of training samples for each epoch with stratified sampling. The sample number for each dataset is 1200, 15000, 1300, 20000, 16000, 8000, 20000, 20000, 10000, 15000, respectively. We fine-tune the model for 1 epochs. The detailed parameters

Table 12: Detailed prompt of GOFA for each dataset in supervised learning.

Dataset	Prompt
Cora-node	*This is a co-citation network focusing on artificial intelligence, nodes represent academic papers and edges represent two papers are co-cited by other papers. What is the most likely paper category for the target paper? Please directly answer the category.*
Cora-link	*This is a co-citation network focusing on artificial intelligence, nodes represent academic papers and edges represent two papers are co-cited by other papers. Is the two target papers co-cited or not? Please only answer yes or no.*
PubMed-node	*This is a co-citation network from Pubmed platform focusing on diabetes mellitus. Nodes represent academic papers and edges represent two papers are co-cited by other papers. What is the most likely paper category for the target paper? Please directly answer the category.*
PubMed-link	*This is a co-citation network from Pubmed platform focusing on diabetes mellitus. Nodes represent academic papers and edges represent two papers are co-cited by other papers. Is the two target papers co-cited or not? Please only answer yes or no.*
Arxiv	*This is a citation network from arxiv platform focusing on the computer science area. Nodes represent academic papers and edges represent citation relationships. What is the most likely paper category for the target Arxiv paper? please directly answer the category.*
WikiCS	*This is a Wikipedia graph focusing on computer science. Nodes represent Wikipedia terms and edges represent two terms have hyperlink. What is the most likely category for this Wikipedia term? Please directly answer the category.*
WN18RR	*This is a knowledge graph from WordNet. Nodes represent an English word and edges represent the relationship between two words. What is the relationship between two target words? Please directly answer the relationship.*
FB15K237	*This is a knowledge graph from freebase. Nodes represent knowledge entities and edges represent relations between two entities. What is the relationship between two target entities? Please directly answer the relationship.*
Products	*This is a co-purchase network from the Amazon platform. Nodes represent the products sold on Amazon and edges represent two products are co-purchased together. What is the most like category for this product? Please directly answer the category.*
ML1M	*This is a recommendation graph from a movie rating platform. Nodes represent user or movie in the platform and edges represent the rating a user gives to a movie. Please predict the user taste to this movie, ranging from 1 to 5.*

for fine-tuning are listed in Table 11. For each dataset, we create a prompt for the LLM decoder to generate the desired answer. In a supervised setting, we ask the LLM model directly to generate the correct answer, instead of doing the selection from the given list. The detailed prompt for each dataset is listed in Table 12. For evaluation, we will match the text output generated by the GOFA with the ground true label to compute the accuracy of the classification task. For the regression task, we will extract the number from the output text and compute the metric with the correct value.