GraphText: Graph Reasoning in Text Space

Jianan Zhao[1,2], **Le Zhuo**[3], **Yikang Shen**[4], **Meng Qu**[1,2], **Kai Liu**[5]
Michael Bronstein[6], **Zhaocheng Zhu**[1,2], **Jian Tang**[1,7,8]
[1]Mila - Québec AI Institute, [2]Université de Montréal, [3]Shanghai AI Lab,
[4]MIT-IBM Watson AI Lab, [5]Division of gRED Computational Science, Genentech Inc.,
[6]University of Oxford, [7]HEC Montréal, [8]Canadian Institute for Advanced Research (CIFAR)

Abstract

Large Language Models (LLMs) have brought transformative changes across various domains by excelling in generative natural language processing. Despite their success, LLMs have not made significant advancements in the realm of graph machine learning. This limitation arises because graphs encapsulate distinct non-Euclidean structures, making it challenging to transform them into natural language that LLMs understand. In this paper, we bridge this gap with a novel framework, GraphText, that translates graphs to natural language. GraphText constructs a graph-syntax tree for each graph, capturing both node features and the complex relationships between nodes. By traversing these trees, a text sequence with structure semantics is produced, enabling LLMs to approach graph reasoning as a text generation problem. Notably, GraphText presents several key benefits. It introduces *training-free graph reasoning*: even without training on graph data, GraphText with ChatGPT can achieve on par with, or even surpass, the performance of supervised-trained graph neural networks through in-context learning. Furthermore, GraphText paves the way for *interactive graph reasoning*, allowing both humans and LLMs to communicate with the model seamlessly using natural language. These capabilities underscore the vast, yet-to-be-explored potential of LLMs in graph machine learning.

1 Introduction

Language stands as a cornerstone of human civilization, acting as the primary medium for knowledge encoding, reasoning, and communication. Large language models (LLMs), pre-trained on extensive text corpora, have showcased remarkable reasoning skills [4, 5]. These LLMs can communicate via natural language both internally [41] and externally with humans or other LLMs [24], demonstrating exceptional skills such as multi-step reasoning [45], decision-making [46, 25], tool use [36], and multi-agent collaboration [30, 19].

Motivation. Despite the remarkable success of LLMs in handling natural languages, their application to other data modalities presents unique challenges, primarily because these data often lack straightforward transformation into sequential text. These challenges are especially severe when dealing with graph-structured data, as different graphs define structure and features in distinct ways. In a parallel vein, graph machine learning methods commonly require the training of specific graph neural networks (GNNs) tailored to individual graphs [20, 38, 44]. Often, GNNs trained on one graph cannot generalize to the unseen structure and feature representations of other graphs. Moreover, the gap between graphs and human languages hinders the application of natural language reasoning to facilitate graph reasoning.

In light of these limitations, a question arises: *can we derive a language for graph in natural language?* In this paper, we give an affirmative answer by proposing to use ***tree*** as an intermediary,

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

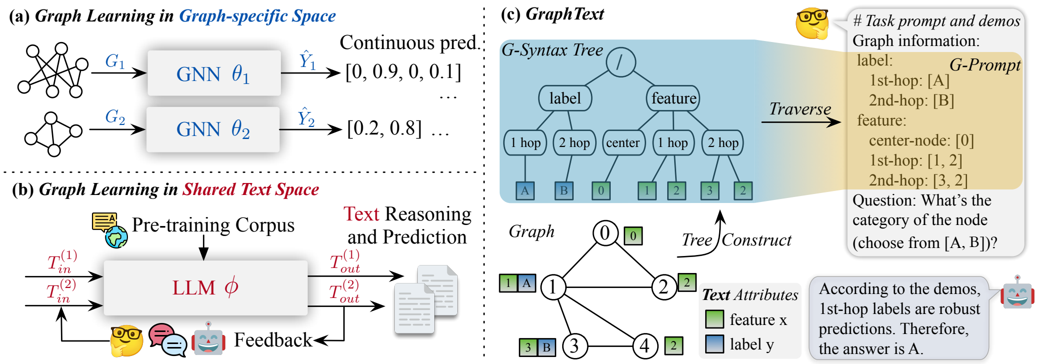


Figure 1: Comparison between **(a)** the GNN framework and **(b)** the proposed GraphText framework. For different graphs G_1 and G_2, different GNNs θ_1 θ_2 are trained to make a graph-specific output prediction in continuous form. In contrast, GraphText encodes the graph information to text sequences $T_{in}^{(1)}$ and $T_{in}^{(2)}$, and generates text reasoning and prediction $T_{out}^{(1)}$ and $T_{out}^{(2)}$ with a graph-shared LLM ϕ. GraphText leverages a pre-trained LLM to perform training-free graph reasoning and enables human and AI interaction for graph reasoning in natural language. **(c)** An example of the GraphText framework that classifies node 0: Given a graph, GraphText constructs a graph-syntax tree that contains both node features (e.g. feature and label) and relationships (e.g. center-node, 1st-hop, and 2nd-hop). Then, GraphText traverses the graph-syntax tree to obtain a sequential text, i.e. graph prompt, and let LLM perform graph reasoning in text space.

elegantly bridging structured data and one-dimensional sequential language. Essentially, *a tree exhibits a hierarchical structure, and traversing it yields a one-dimensional sequence.* On top of that, as shown in Figure 1 (c), we propose a novel framework GraphText, which takes graph data to build a graph-syntax tree. Traversing it results in a graph prompt expressed in natural language, allowing an LLM to approach graph reasoning as a text-generation task.

Main contributions. First, GraphText serves as a flexible and general framework for graph reasoning, effectively preserving structural semantics into the graph prompt. Moreover, it also allows incorporate inductive bias of GNNs, such as feature propagation and feature similarity-based propagation, by constructing different graph-syntax trees. Second, we show that GraphText enables effective **training-free graph reasoning**. GraphText with ChatGPT sets a new benchmark in graph in-context learning, outperforming the best ICL baseline by an average of 34.0%. Remarkably, even without training on graph data, GraphText can deliver performance on par with, or even surpass, supervised graph neural networks through in-context learning. To our best knowledge, this is the first time that a language model with ICL outperforms supervised GNN. This highlights the vast potential of foundation models in the realm of graph machine learning. Third, GraphText fosters **interactive graph reasoning**: With its capacity to generate and **explain** predictions in natural language, humans can directly engage with GraphText. As shown in Figure 4 (b), through interactions with humans and other LLMs, GraphText refines its graph reasoning capabilities.

2 Methodology

In this section, we present how GraphText performs graph reasoning in text space. Out of the three fundamental problems of graph ML (graph classification, node classification, and link prediction), we take node classification as an example to introduce our idea. We however note that our discussion applies to other graph tasks.

2.1 The GraphText Framework

Let us be given an attributed graph $G = (V, E, \boldsymbol{X})$ with nodes V and edges E, whose structure is represented as the $|V| \times |V|$ adjacency matrix $\boldsymbol{A}$ and node features as the $|V| \times d$ feature matrix $\boldsymbol{X}$. Given a subset $L \subset V$ of labeled nodes with labels Y_L, the goal of node classification is to predict the labels Y_U of the unlabeled nodes $U = V \setminus L$. Graph Neural Networks (GNNs) are the standard architecture for such problems. As shown in Figure 1 (a), a GNN directly learns a parametric map

$$\hat{y}_i = f_{\text{GNN}}(G; \theta_G)_i \tag{1}$$

between the input graph $G \in \mathcal{G}$ and the output labels $\hat{Y} \in \mathcal{Y}$, assigning to each node i its predicted label $\hat{y}_i$. The training of GNN attempts to find parameters θ_G such that $\hat{y}_i \approx y_i$ on the training set. Note that standard GNNs are ***graph-specific*** functions, i.e. $f_{\text{GNN}}(\cdot; \theta_G) : G \mapsto \hat{Y}$, which do not generalize to other graphs, since other graphs $G' \in \mathcal{G}$ define distinct distributions of Y', $\boldsymbol{A}'$, and $\boldsymbol{X}'$, or even different types of features such as continuous, categorical, or text features.

To solve the generalization problem mentioned above, we propose to perform graph reasoning as a *text-to-text* problem [32], as shown in Figure 1 (b). Inspired by prompt tuning [4, 26], we construct two graph-specific maps to form the input and output space of a text-to-text problem: a map $g : G \mapsto T_{in}$ that maps the graph input to text space, and a map $h : T_{out} \mapsto \tilde{Y}$ that maps the output of LLM to label predictions $\tilde{Y}$. In this way, we can use a generative large language model f_{LLM} to perform graph reasoning as

$$\tilde{y}_i = h(f_{\text{LLM}}(g(G)_i; \phi)), \tag{2}$$

where $g(G)_i = T_{in}[i]$ denotes the text sequence representing node i. Different from GNNs, $f_{\text{LLM}}(\cdot; \phi) : \mathcal{T} \to \mathcal{T}$ is a ***graph-shared*** function, where both input and output are in text space, i.e. $T_{in}, T_{out} \in \mathcal{T}$, which not only activates of parametric knowledge encoded in the model $f_{\text{LLM}}(\cdot; \phi)$, but also enables natural-language-interactions between human and AI agents to facilitate graph reasoning.

Specifically, as node classification, link prediction and graph classification are essentially classification tasks, we can naturally formulate these graph reasoning tasks as multi-choice QA problem [35] and design h as the map from predicted choice $T_{out} \in \mathcal{T}$ to the corresponding prediction $\tilde{Y}$. However, the design of g that maps the structural graph information into the text space of natural language is still a non-trivial problem.

The primary challenge in converting graph data to language lies in handling its relational structure, which fundamentally deviates from the one-dimensional sequential nature of text data. Inspired by the syntax trees in computation linguistics which describe the syntactic structure and semantics between nodes (words), we introduce graph-syntax trees as a bridge between relational and sequential data. The traversal of such a tree produces a sentence in natural language, which is fed to LLM for graph reasoning.

Specifically, as shown in Figure 1 (c), we compose a graph-syntax tree consisting of two types of nodes: leaf nodes that contain the graph *features*, and the internal nodes that describe the *relational semantics* of the leaf nodes. Next, we describe how to extract graph information as feature and relational information in Section 2.2, and how to build a graph-syntax tree in Section 2.3.

2.2 Feature and Relation Extraction for Graph-Syntax Tree

In this section, we discuss how to prepare the graph information consisting of feature and relation information to serve as the leaf and internal nodes for a graph-syntax tree. For text features that serve as leaf nodes in graph-syntax tree, GraphText constructs a text feature set $F \in \mathcal{T}$ for a graph $G \in \mathcal{G}$ (with or without text features) composed of multiple types of features for each node, e.g. feature and label, in natural language. Specifically, for each node v_i and feature type m, we construct a text sequence $\text{F}_m[i]$ in natural language:

$$\text{F}_m[i] = \{t_1, t_2, \cdots t_{l_m}\}, \;\; \text{F}_m[i] \in \mathcal{T}, \tag{3}$$

where the sequence is of length l_m. Each text feature F_m can be derived from either sequential text features or continuous features. For text-attributed graphs, the text features can be directly added to the text features F. For example, we can directly add the text sequences of "title" and "abstract" into F for citation graphs. For continuous features, e.g. the raw feature $\boldsymbol{X}$ or other graph embeddings, we propose to use discretization methods, e.g. clustering, to transform the continuous feature into a discrete space and then derive sequential data from it. For simplicity, we use the cluster index of K-means to generate a sequence of length 1 for all continuous features as K-means is effective in our experiments.

For relational information that serves as internal nodes in a graph-syntax tree, GraphText derives a set of matrices R where each $\boldsymbol{R}_n \in R$ is a $|V| \times |V|$ matrix, depicting one type of relationship between nodes. Choices of $\boldsymbol{R}_n$ may be the original graph [20], high-order connectedness [43], page-rank matrices [21], or any matrices that encode node-pair information. These relationships play an important role in determining the nodes and structure of the graph-syntax tree, which further improves the graph reasoning ability of LLMs.

2.3 Graph-syntax Tree Composition

We now discuss how to build a prompt from the graph text feature F and relationships R using a graph-syntax tree. Formally, a graph-syntax tree is an ordered tree: a directed acyclic graph (DAG) with nodes $\tilde{T} \in \mathcal{T}$ and edges $\tilde{E}$. In a graph-syntax tree, e.g. the one in Figure 1 (c), each node stores a *text sequence* in natural language, where the root node is an empty node; the leaf nodes $\tilde{T}_{\mathrm{L}}$ are text sequences in the graph text features, i.e. $\forall\, \mathrm{T}_i \in \tilde{T}_{\mathrm{L}}, \mathrm{T}_i \in F$; the internal nodes $\tilde{T}_{\mathrm{I}}$ are text sequences in natural language, i.e. $\forall\, \mathrm{T}_i \in \tilde{T}_{\mathrm{I}}, \mathrm{T}_i \in \mathcal{T}$. A graph syntax tree is constructed in three steps: (1) construct an ego-subgraph [17] G_i for target node v_i based on relationship R (2) select leaf nodes $\tilde{T}_{\mathrm{L}}$ based on the relationship R. (3) build up internal nodes $\tilde{T}_{\mathrm{I}}$ and edges $\tilde{E}$ based on the leaf nodes' types and their relationship with the graph[1]. Notably, the leaf nodes are sorted according to their relationships with the center-node, preserving the relative relationship in a one-dimensional order.

We illustrate this with a node classification example shown in Figure 1 (c). Before building the graph-syntax tree, GraphText determines the text features composed of raw features and observed labels, i.e. $F = \{\mathrm{F}_X[i], \mathrm{F}_Y[i] \mid \forall v_i \in V\}$, and a relationship set composed of shortest path distance (SPD)-based relations: center-node, 1st-hop, and 2nd-hop, i.e. $R = \{\boldsymbol{R}_{\text{SPD=0}}, \boldsymbol{R}_{\text{SPD=1}}, \boldsymbol{R}_{\text{SPD=2}}\}$. Then, for target node v_i (0 in the example), an ego-subgraph [17] (with nodes [0,1,2,3,4]) is sampled based on the relative relationship between v_i and other nodes. Finally, a graph-syntax tree is constructed with leaf nodes $\tilde{T}_{\mathrm{L}} = \{\mathrm{F_X}[0], \mathrm{F_X}[1], \mathrm{F_X}[2]\mathrm{F_X}[3], \mathrm{F_X}[4], \mathrm{F_Y}[1], \mathrm{F_Y}[3]\}$, the internal nodes $\tilde{T}_{\mathrm{I}}$ ={"center-node", "1st-hop", "2nd-hop", "label", "feature"}, and the corresponding edges. The traversal of the resulting graph-syntax tree leads to a text sequence in natural language.

Compared with the direct flattening of (sub)graphs [39, 8, 13], using a graph-syntax tree-based prompt has many advantages: Above all, unlike a graph, which has no topology order, a syntax tree is a DAG that can be topologically sorted, which gracefully converts a relational structure to a text sequence. Moreover, GraphText easily incorporates the inductive biases of GNNs through the construction of node text features F and relationships R. For example, we can easily encode the message-passing prior using the propagated feature $\boldsymbol{A}^k\boldsymbol{X}$ [49] as the node features F. We can also incorporate the feature similarity-based aggregation [38] by adding $\boldsymbol{X}\boldsymbol{X}^\top$ to R. These graph-based inductive biases can significantly boost LLMs' graph reasoning performance (s Section A.2). Last but not least, a tree naturally defines a hierarchical structure, which LLMs are proficient in reasoning on [25], by training on code data [6] and web page data [37].

3 Experiments

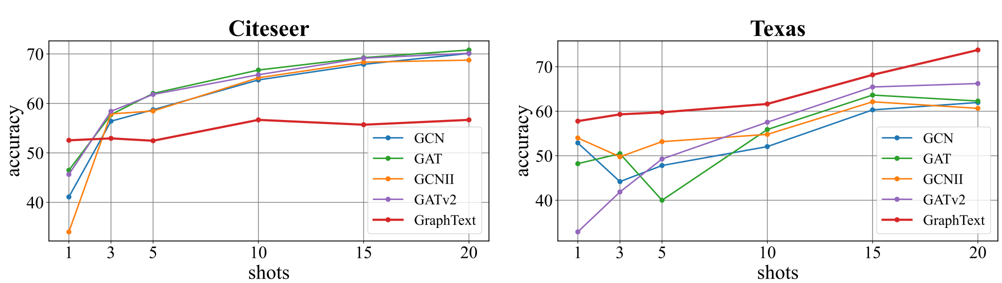


Figure 2: Few-shot in-context learning node classification accuracy. We perform 1, 3, 5, 10, 15, and 20-shot node classification on Citeseer and Texas datasets.

[1]The hierarchy of the tree can be defined flexibly, but we have empirically discovered that a simple configuration, with attribute type at the top hierarchy and relation type at the bottom hierarchy for internal nodes, as illustrated in Figure 1 (c), yields strong performance. Details are available in Section A.2.

Table 1: Node classification results (accuracy %). Δ GCN and Δ Best ICL denotes the aboslute performance gain of GraphText over GCN and ICL baselines, respectively.

Setting	Model	Cora	Citeseer	Texas	Wisconsin	Cornell
Supervised Learning	GCN	81.4	69.8	59.5	49.0	37.8
	GAT	80.8	69.4	54.1	49.0	**45.9**
	GCNII	81.2	69.8	56.8	51.0	40.5
	GATv2	**82.3**	**69.9**	**62.2**	**52.9**	43.2
In-Context Learning	NeighborText	26.3	13.7	5.4	9.8	21.6
	GML	38.5	28.4	10.8	23.5	21.6
	GraphML	49.9	28.9	16.2	33.3	29.7
	GraphText of+or	33.4	36.9	5.4	29.4	24.3
	GraphText of+sr	52.1	50.4	73.0	60.8	46.0
	GraphText sf+or	64.5	51.0	73.0	35.3	48.7
	GraphText sf+sr	**68.3**	**58.6**	**75.7**	**67.6**	**57.9**
Comparisons	Δ SFT-GCN	-13.1%	-11.2%	+16.2%	+18.6%	+20.1%
	Δ Best ICL	+18.4%	+29.7%	+59.5%	+34.3%	+28.2%

We conduct extensive experiments to demonstrate the effectiveness of GraphText. Firstly, in this Section, we delve into the remarkable capacity of GraphText for training-free graph reasoning. In the appendix, we provide more detailed results, including: Section A.1 highlights the interactive graph reasoning capabilities of GraphText with a real-world graph reasoning example. We further analyze various ablations of graph-syntax trees in Section A.2. Besides, Section A.3 illustrates how GraphText can seamlessly function as a versatile framework, catering to both in-context learning and instruction tuning across on both general graph and text-attributed graphs.

One unique ability of GraphText is the training-free graph reasoning by in-context learning (ICL). In this section, we demonstrate this capability on the node classification tasks. Specifically, we use two citation datasets (*Cora* [28] and *Citeseer* [15]), and three webpage datasets (*Texas*, *Wisconsin*, and *Cornell* [31]). We compare GraphText with two types of baselines: (1) *Supervised learning GNNs*: including *GCN* [20] and *GAT* [38], along with their more recent variants *GCNII* [7] and *GATv2* [3]; (2) *In-Context Learning (ICL) LLM* baselines: NeighborText [8] (the only available raw text features in the graph is the observed node labels), GML, and GraphML [16]. The GNN baselines are supervised trained solely for inference on one dataset. In contrast, ICL baselines utilize a single pre-trained LLM (ChatGPT) for all datasets without any graph-specific training. For GraphText, we derive four ablations based on the text and relation information used when constructing the graph-syntax tree. For *feature information*, the original graph information (feature, label) and synthetic text information (propagated feature/labels) are denoted as *of* and *sf*. For *relation information*, the original graph relation and the synthetic relations (e.g. shortest-path based, feature similarity-based, and personalized PageRank relations), denoted as *or* and *sr*, respectively.

The experimental findings, as shown in Table 1 and Figure 2, reveal that methods using the raw graph as a prompt, such as NeighborText, GML, GraphML, and GraphText of+or, exhibit poor performance. However, the incorporation of graph inductive bias into both synthetic text features and relations leads to significant performance improvements across all datasets. Notably, GraphText sf+sr, integrating these elements, sets a new benchmark in graph in-context learning, outperforming the best existing ICL baselines by an average of 34.0%. Impressively, GraphText occasionally **exceeds supervised learning GNN baselines** through in-context learning, especially in scenarios of low label rates (as indicated in Figure 2) and within heterophilic datasets. The remarkable efficacy of GraphText in training-free graph reasoning underscores the immense potential of LLMs in the realm of graph learning.

4 Conclusion

In this paper, we propose GraphText, a language for graph that utilizes a graph-syntax tree to seamlessly convert graph information into a structured prompt, enabling an LLM to perform training-free graph reasoning. GraphText delivers performance on par with, or even surpasses, supervised graph neural networks through in-context learning. What's more, GraphText fosters explainable and interactive graph reasoning in natural language which enables humans and LLMs to engage with graph learning interactively. These remarkable abilities highlight the immense and untapped potential of LLMs in the realm of graph machine learning.

References

[1] Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernández Ábrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan A. Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vladimir Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, and et al. Palm 2 technical report. *CoRR*, 2023.

[2] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. Scaling graph neural networks with approximate pagerank. In *KDD*, 2020.

[3] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *ICLR*, 2022.

[4] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[5] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Túlio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with GPT-4. *CoRR*, 2023.

[6] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, 2021.

[7] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *ICML*, 2020.

[8] 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. *CoRR*, 2023.

[9] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *ICLR*, 2021.

[10] Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S. Dhillon. Node feature extraction by self-supervised multi-scale neighborhood prediction. In *ICLR*, 2022.

[11] David Dohan, Winnie Xu, Aitor Lewkowycz, Jacob Austin, David Bieber, Raphael Gontijo Lopes, Yuhuai Wu, Henryk Michalewski, Rif A. Saurous, Jascha Sohl-Dickstein, Kevin Murphy, and Charles Sutton. Language model cascades. *CoRR*, 2022.

[12] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *ICLR*, 2022.

[13] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models, 2023.

[14] Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. In *ICLR*, 2023.

[15] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pages 89–98, 1998.

[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. *CoRR*, abs/2305.15066, 2023. doi: 10.48550/arXiv.2305.15066. URL `https://doi.org/10.48550/arXiv.2305.15066`.

[17] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

[18] Xiaoxin He, Xavier Bresson, Thomas Laurent, and Bryan Hooi. Explanations as features: Llm-based features for text-attributed graphs. *CoRR*, abs/2305.19523, 2023. doi: 10.48550/arXiv.2305.19523. URL `https://doi.org/10.48550/arXiv.2305.19523`.

[19] Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, and Chenglin Wu. Metagpt: Meta programming for multi-agent collaborative framework, 2023.

[20] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[21] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*, 2019.

[22] Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. In *NeurIPS*, 2021.

[23] Chaozhuo Li, Bochen Pang, Yuming Liu, Hao Sun, Zheng Liu, Xing Xie, Tianqi Yang, Yanling Cui, Liangjie Zhang, and Qi Zhang. Adsgnn: Behavior-graph augmented relevance modeling in sponsored search. In Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai, editors, *SIGIR*, 2021.

[24] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: communicative agents for "mind" exploration of large scale language model society. *CoRR*, 2023.

[25] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *ICRA*, 2023.

[26] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 2023.

[27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL `https://openreview.net/forum?id=Bkg6RiCqY7`.

[28] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

[29] OpenAI. Introducing chatgpt. 2023. URL `https://openai.com/blog/chatgpt`.

[30] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. *CoRR*, 2023.

[31] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *ICLR*, 2020.

[32] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 2020.

[33] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash, editors, *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 3505–3506. ACM, 2020. doi: 10.1145/3394486.3406703. URL `https://doi.org/10.1145/3394486.3406703`.

[34] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.

[35] Joshua Robinson and David Wingate. Leveraging large language models for multiple choice question answering. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL `https://openreview.net/pdf?id=yKbprarjc5B`.

[36] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *CoRR*, 2023.

[37] 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.

[38] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

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

[40] Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, and Jifeng Dai. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *CoRR*, abs/2305.11175, 2023. doi: 10.48550/arXiv.2305.11175. URL `https://doi.org/10.48550/arXiv.2305.11175`.

[41] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

[42] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*, 2019.

[43] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *ICML*, 2018.

[44] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[45] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *CoRR*, 2023.

[46] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *ICLR*, 2023.

[47] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. Natural language is all a graph needs. *CoRR*, abs/2308.07134, 2023. doi: 10.48550/arXiv.2308.07134. URL `https://doi.org/10.48550/arXiv.2308.07134`.

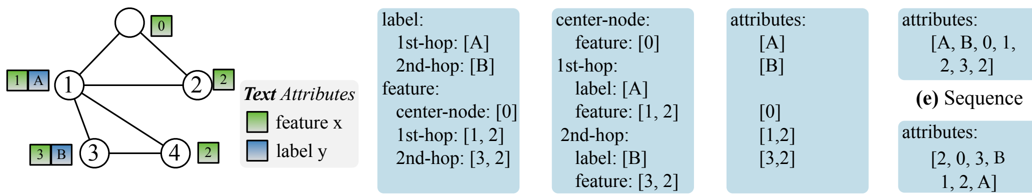
[48] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation? In *NeurIPS*, 2021.

[49] Wentao Zhang, Ziqi Yin, Zeang Sheng, Yang Li, Wen Ouyang, Xiaosen Li, Yangyu Tao, Zhi Yang, and Bin Cui. Graph attention multi-layer perceptron. In *KDD*, 2022.

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

Table 2: Interactive graph reasoning results (accuracy %) on Cora (node # 2188). The table showcases the performance of GPT-4 and ChatGPT before and after human interactions with 15 times of evaluation. The reasoning patterns include PPR, Center-node, and instances where the model was Confused to respond or Refused (Conf./Ref.) to make their reasoning/prediction. See Figure 4 (c) and Appendix E for details.

Model	Interaction	Accuracy	Reasoning		
			PPR	Center-node	Conf./Ref.
GPT-4	Before	73.3	73.3	26.7	0
	After	**100 (+26.7)**	100	0	0
ChatGPT	Before	26.7	26.7	53.3	20.0
	After	**63.6 (+36.9)**	72.7	18.2	9.1



(a) An example graph **(b)** GraphText **(c)** Rev. hierarchy **(d)** w/o struct-sem **(e)** Sequence **(f)** Set

Figure 3: Ablations of graph-syntax trees. **(a)** An example graph. **(b)** GraphText prompt text (the full example can be found in Figure 1). **(c-f)** prompts text of different tree designs.

A More Experiments

A.1 Interpretable and Interactive Graph Reasoning

In this section, we illustrate that GraphText facilitates effective **interactive graph reasoning**: through its ability to generate and **clarify** predictions in natural language, both humans and LLMs can directly interact with GraphText.

To illustrate this concept, we will use Cora node #2188 as an example. Figure 4 (a) shows two types of text features we use: the center-node pseudo labels and the PPR (Personalized PageRank) pseudo label sequence, where the first PPR neighbor denotes the most important label prediction. Upon examining the demonstrations (marked in blue), it becomes apparent that the PPR pseudo-labels are more robust than center-node for label prediction. However, it is commonly believed that center-nodes are the most important for label prediction. Hence, this is a challenging example for LLM to update their previous preference of center-node-based prediction to PPR-based graph reasoning. We leverage GraphText with ChatGPT and GPT-4 and summarize theis reasoning processes and results are illustrated in Figure 4 and Table 2 respectively, from which we draw several key insights:

1. LLMs inherently possess the knowledge and inductive bias toward graph reasoning. Specifically, both ChatGPT and GPT-4 acknowledge the importance of center-nodes and sometimes make predictions based on center-node labels. ChatGPT exhibits reasoning with the center-node bias 53.3% of the time, while GPT-4 does so at a rate of 26.7%.

2. LLMs can update their prior inductive bias based on demonstrations. Through in-context learning, GraphText can recalibrate their bias and make more accurate predictions. Our observations indicate that GPT-4 significantly outperforms ChatGPT, achieving an accuracy of 73.3%, markedly superior to ChatGPT's 26.7%.

3. LLMs can adapt their prior inductive bias based on human feedback. Figure 4 (b) provides an illustrative example, with a detailed reasoning of LLM can be found in Appendix E. Specifically, after a human interaction that reminds LLM of what PPR is and reevaluate the prediction, GPT-4 shows remarkable adaptability, achieving an impeccable accuracy of 100% and adhering to the PPR logic. Meanwhile, ChatGPT also enhances its performance notably (gaining 36.9% in accuracy), but occasionally maintains its antecedent biases.

Table 3: Ablations of GraphText on Cora and Citeseer.

Model	Cora		Citeseer	
	Acc. %	Δ	Acc. %	Δ
GraphText	**68.3**	-	**58.6**	-
rev. hierarchy	68.3	-0 %	57.6	-1.7 %
w/o struct-sem	67.8	-0.5 %	56.3	-3.9 %
sequence	67.0	-1.3 %	53.0	-9.6 %
set	65.9	-2.4 %	56.4	-3.8 %

In summary, through graph reasoning in natural language, GraphText can effectively leverage its pre-trained knowledge to engage in graph reasoning and, crucially, adapt its existing knowledge through demonstrations or external feedback.

A.2 Ablation Studies on Graph-syntax Trees

The graph-syntax tree serves as the core design of GraphText, transforming a graph into a one-dimensional natural language sequence. Within GraphText, the text feature and relational information of a graph is initially formulated, followed by the construction of a graph-syntax tree. This section delves into the ablations of various methods for building graph-syntax trees.

As shown in Figure 3, besides the proposed GraphText method for constructing a graph-syntax tree, we present four ablation types: (1) Reverse hierarchy (denoted as **rev. hierarchy** in Figure 3 (c): The tree hierarchy is inverted, positioning the relationship type at the top and the text feature type at the bottom. (2) Without structure semantics (denoted as **w/o struct-sem** in Figure 3 (d)): The internal nodes of the graph-syntax tree are eliminated, losing the structural semantics, but the GraphText hierarchy remains intact. (3) Sequential prompt (denoted as **sequence** in Figure 3 (e)): The tree hierarchy is removed, yielding a sequence of text features. (4) Set prompt (denoted as **set** in Figure 3 (f): Sequence order is removed, yielding a set.

From Table 3, several observations can be made: (1) The graph-syntax tree of GraphText consistently outperforms the other ablations, underscoring the efficacy of our design. (2) The hierarchical structure of the tree plays a crucial role in the design of the graph prompt. Specifically, we observe a significant performance drop when using a *sequence* or a *set* to represent the graph information. (3) Variations in the tree hierarchy design can impact performance; for instance, *rev. hierarchy* underperforms GraphText on Citeseer. (4) The comparison between *w/o struct-sem* and GraphText reveals the importance of making LLMs aware of the structure semantics. This suggests that LLMs utilize the structure semantics in the prompt to distinguish and understand different attributes during graph reasoning.

A.3 Experiments on Text-attributed Graphs

In this section, we demonstrate that GraphText is also applicable to text-attributed graphs. As depicted in Table 4, we conducted training-free node classification on the Cora and Citeseer datasets with both raw text features [8] and continuous features [34]. We observed that using closed-source LLMs, such as ChatGPT, the performance lags behind the GNN baseline methods. Thus, we further explored the potential of instruction tuning on currently available open-source LLMs, such as LLaMA-2 [37]. For natural language prompts construction. Specifically, we freeze the pre-trained LLaMA-2 parameters and expand the original vocabulary of LLaMA-2 by introducing selected options as new tokens and fine-tune an MLP encoder for mapping continuous features to discrete text space.

From the results in Table 4, it is evident that even with a relatively smaller open-source model, LLaMA-2-7B, our best results from instruction tuning across various settings surpass those of ChatGPT and approach the GNN baselines. This validates that our method can be beneficial in an instruction-tuning scenario. It also implies that using GraphText, we can feasibly fine-tune smaller

Table 4: Node classification results (accuracy %) on real-world text-attributed graphs. Experiments are conducted using in-context learning with ChatGPT, as well as instruction tuning with LLaMA-2-7B. Note that "text" refers to raw text features, while "feat" represents the continuous features on the graph. The top results for each category are highlighted in bold.

Framework	Model	Cora	Citeseer
GNNs	GCN	89.13	74.92
	GAT	**89.68**	**75.39**
GraphText-ICL	ChatGPT-text	**67.77**	**68.98**
	ChatGPT-feat	10.68	16.14
	ChatGPT-feat+text	65.19	66.46
GraphText-SFT	LLaMA-2-7B-text	60.59	49.37
	LLaMA-2-7B-feat	**87.11**	**74.77**
	LLaMA-2-7B-feat+text	77.53	73.83

open-source LLMs with reasonable computational costs, achieving performances that can rival or even surpass those of much larger closed-source models, such as ChatGPT or GPT-4.

Another intriguing observation is the notably poor performance of ChatGPT in settings incorporating continuous features - nearing a random guess. This is attributable to the inherent limitation of these closed-source LLMs: they are designed to process raw discrete text inputs and fail to directly handle the continuous inputs. In contrast, open-source LLMs possess the ability to map these continuous embeddings into their embedding space, facilitating improved performance.

Upon contrasting these two groups of models, we noticed a decline in the performance of open-source models when processing raw text inputs. This decline can be ascribed to the constraints imposed by the size of the LLM parameters and the volume of pre-training corpora used. It suggests that harnessing larger-scale open-source models, such as LLaMA-2 variants including 13B, 30B, and 70B, would significantly bolster their modeling capacity for raw text. Concurrently, by leveraging the ability to process continuous embeddings, these models would inevitably exhibit enhanced graph reasoning capabilities, paving the way for more sophisticated graph-based applications.

B Connections to Other Frameworks

Unlock Graph Space for Language Models. Large Language Models (LLMs) [4, 29, 1, 5] possess impressive reasoning capabilities [41, 45, 14]. At the heart of LLMs' reasoning prowess is their ability to process and generate natural language inputs and outputs, enabling flexible interactions [11] with both humans and AI agents. This unique capability empowers them with remarkable abilities such as complex reasoning [14] and decision-making [46, 25]. Despite their success, applying LLMs to relational graph data remains challenging, primarily due to the absence of a natural language representation for graphs. GraphText bridges this gap by providing a novel framework that enables LLMs to seamlessly integrate and reason over relational graph data using the same natural language capabilities, thereby unlocking their potential for a wide range of graph-based applications.

Training-free Graph Reasoning. Graph neural networks (GNNs) [20, 44] excel in handling relational graph data, thanks to the message-passing mechanism for aggregation and transformation of neighborhood representations. Their standout performance can be attributed to their intrinsic capability to assimilate graph inductive biases. This incorporation of inductive biases can be achieved by designing input features with the graph structure prior, such as position embeddings [12, 48, 22] and propagated features [42, 49]. Besides, the graph inductive bias can also helps the design of message passing, such as feature similarity-based message passing [38] or high-order aggregation [21, 2, 9]. However, as highlighted in Section 2.1, due to the variance in both structure and feature, the majority of GNNs are *graph-specific*, which can not transfer to unobserved structures and features.

In a parallel vein, GraphText also taps into the potent ability to infuse graph inductive biases for graph reasoning, achieved through designing both the feature and relation information for the graph-syntax tree. Setting itself apart from GNNs, GraphText approaches graph reasoning in a *graph-*

shared domain, facilitating the broader applicability of a single LLM to diverse graphs and offering training-free and interactive graph reasoning.

Connecting Both Worlds. Recent endeavors [10, 50, 18] have aimed to merge the language and graph domains. Most methods involve transitioning the problem into a graph-specific realm, utilizing a combination of a text-encoder (either pre-trained [10] or learned [23]) and a GNN predictor. This methodology still falls into a *graph-specific* paradigm, as shown in Figure 1 (a), where a model is trained for a specific graph. Very recently, concurrent works [16, 47, 39, 8, 13] exploring to leverage LLMs for graph-related tasks. These methods propose to describe the nodes, features and edge information of a graph with a prompt template. Nevertheless, the structural information is directly flattened into an unstructured prompt, e.g. a list of connected edges around the center node, which leads to poor graph reasoning capability for LLMs which is far worse than GNNs' performance [39, 8]. Moreover, most of these works are limited to text-attributed graphs where a text description is required for each node, which limits the application of LLMs on other types of graphs.

In contrast, GraphText is fundamentally different from these works. Foremost, GraphText proposes a language defined by a graph-syntax tree, offering a flexible and structured approach for seamlessly integrating graph inductive biases that incorporates structure syntax inside the graph-syntax tree. Moreover, it also serves as a general framework for graph reasoning, which can be applied to scenarios encompassing in-context learning and instruction tuning. It accommodates various types of graphs, including general graphs and text-attributed graphs, and is adaptable to both closed-source LLMs [29, 5] and open-source LLMs [37].

C Experimental Settings

C.1 Implementation Details

In our experiments node classification is approached as a multi-choice QA problem, and we employ the prompt detailed in Appendix D. The raw text features can be directly leveraged to form the textual information F. Additionally, we utilize the dataset metadata to extract raw text labels, creating a textual feature for each node. Nodes without data are assigned the value "NA". Consequently, every general graph can be perceived as a text-attributed graph with a minimum of one type of text feature, namely the label. For continuous features, we use K-means to discretize continuous features (with the K being the number of classes).

During the in-context-learning experiments, to prevent meaningless demonstrations lacking neighboring labels, we choose a sample with the highest degree from each label set. For the instruction tuning experiments using open-source LLMs, we can leverage continuous features in a more flexible way. Specifically, inspired by multi-modality LLMs [40], we use a Multi-Layer Perceptron (MLP) projector to map continuous features into the input text space, the token embedding space of LLaMA-2 [37]. We utilize AdamW [27] in conjunction with DeepSpeed [33] to train the huggingface LLaMA2-7b model [2], with FP16 activated.

C.2 Datasets

Table 5: The statistics of the datasets.

Benchmarks	#Nodes	#Edges	#Classes	#Features	#Train	#Validation	#Test
Cora	2708	5278	7	1433	140	500	1000
Cora-TAG	2708	5278	7	1433	1624	541	543
Citeseer	3327	4552	6	3703	120	500	1000
Citeseer-TAG	3327	4552	6	3703	1911	637	638
Cornell	183	298	5	1703	87	59	37
Texas	183	325	5	1703	87	59	37
Wisconsin	251	515	5	1703	120	80	51

[2] https://huggingface.co/meta-LLaMA/LLaMA-2-7b-hf

In this section, we provide more relevant details about the datasets we used in experiments. The dataset statistics are provided in Table 5. The datasets can be categorized into citation network datasets (i.e. Cora, Citeseer, and ogbn-arxiv) and web-page networks (i.e. Cornell, Texas, and Wisconsin), additionally, we use two text-attributed-graph (TAG) version of Cora and Citeseer, denoted as Cora-TAG and Citeseer-TAG.

Citation graphs: Most GNN-related studies, as referenced in works like [20, 38], often employ citation networks as benchmarking tools. Within these networks, nodes represent papers from the computer science domain. The features of these nodes are derived from bag-of-word vectors of the respective paper titles. Edges depict the citation links between these papers, while labels indicate the paper's specific categories. The text features are the title and abstract of the paper.

WebKB graphs [31]: Sourced by Carnegie Mellon University, aggregates web pages from computer science departments across several universities. We employ three specific subsets from this collection: Cornell, Texas, and Wisconsin. In these subsets, each node symbolizes a web page, while edges denote hyperlinks connecting them. Nodes are characterized by a bag-of-words representation derived from their respective web pages. These pages have been meticulously categorized into five distinct classes: student, project, course, staff, and faculty.

The datasets mentioned above can be found in the following URLs: Cora [3], Citeseer [4], Cora-TAG [5], Citeseer-TAG [6], Texas [7], Cornell [8], Wisconsin [9].

C.3 Hyperparameters

In GraphText, the selection of text features F and relations R are the most important parameters. Here we discuss their choices and report the selected parameters in Table 6. For text features F, there are several choices: propagated features $\boldsymbol{A}^k\boldsymbol{X}$, propagated labels $\boldsymbol{A}^k\boldsymbol{Y}_L$, raw features $\boldsymbol{X}$ and labels $\boldsymbol{Y}_L$; For relationship R, there are several choices: k-hop shortest-path distance, denoted as $\boldsymbol{S}_k$, propagated feature similarity, denoted as $\text{sim}(\boldsymbol{A}^k\boldsymbol{X})$, and pagerank matrix [21], with restart probability $\alpha = 0.25$, denoted as $\boldsymbol{\Pi}$.

Table 6: GraphText in-context learning hyperparameters.

	text features	**Relations**
Cora	$\boldsymbol{A}^2\boldsymbol{Y}_L$,$\boldsymbol{A}^3\boldsymbol{Y}_L$	$\boldsymbol{S}_0$,$\boldsymbol{\Pi}$, $\text{sim}(\boldsymbol{A}^2\boldsymbol{X})$,$\text{sim}(\boldsymbol{A}^3\boldsymbol{X})$
Citeseer	$\boldsymbol{X}$, $\boldsymbol{A}^3\boldsymbol{Y}_L$	$\boldsymbol{S}_0$,$\boldsymbol{S}_2$,$\boldsymbol{\Pi}$, $\text{sim}(\boldsymbol{A}^2\boldsymbol{X})$
Texas	$\boldsymbol{A}^2\boldsymbol{Y}_L, \boldsymbol{A}^3\boldsymbol{Y}_L$	$\boldsymbol{S}_2$
Wisconsin	$\boldsymbol{X}, \boldsymbol{Y}_L$	$\text{sim}(\boldsymbol{X})$, $\boldsymbol{S}_3$
Cornell	$\boldsymbol{A}\boldsymbol{Y}_L, \boldsymbol{A}^4\boldsymbol{Y}_L$	$\boldsymbol{S}_1$,$\text{sim}(\boldsymbol{A}^3\boldsymbol{X})$

D Prompt Examples

D.1 Few-shot In-context Learning

Example of Citeseer:
[Human]: You are a helpful assistant that classifies the topic of an academic paper based on the labels of the cited papers. You are going to choose the correct answer from several choices of paper categories: [A: Agents, B: Artificial Intelligence, C: Database, D:

[3] https://relational.fit.cvut.cz/dataset/CORA
[4] https://linqs.soe.ucsc.edu/data
[5] https://github.com/CurryTang/Graph-LLM
[6] https://github.com/CurryTang/Graph-LLM
[7] https://docs.dgl.ai/en/0.8.x/api/python/dgl.data.html#node-prediction-datasets
[8] https://docs.dgl.ai/en/0.8.x/api/python/dgl.data.html#node-prediction-datasets
[9] https://docs.dgl.ai/en/0.8.x/api/python/dgl.data.html#node-prediction-datasets

Information Retrieval, E: Machine Learning, F: Human Computer Interaction]

Here are a few examples:
<information>
<third-order_pseudo_labels>
<center_node>['A']</center_node>
<1st_feature_similarity_graph>['A', 'A', 'A']</1st_feature_similarity_graph>
<ppr>['A', 'B', 'A']</ppr>
</third-order_pseudo_labels>
</information>
<question>What's the topic of academic paper given the information above?</question>
<answer>A</answer>

Remaining examples ...

Now let's answer the question below:
<information>
<third-order_pseudo_labels>
<1st_feature_similarity_graph>['C', 'B', 'B']</1st_feature_similarity_graph>
<ppr>['C']</ppr>
</third-order_pseudo_labels>
</information>

What's the topic of the paper given the information above? Valid choices are [A: Agents, B: Artificial Intelligence, C: Database, D: Information Retrieval, E: Machine Learning, F: Human computer interaction]. Remember, your answer should be in the form of the class choice wrapped by <answer></answer>.

[Assistant]: <answer>C</answer>

D.2 Instruction Tuning

Example of Cora:
[Human]: Your goal is to perform node classification. You are given the information of each node in a xml format. Using the given information of a node, you need to classify the node to several choices: [<c0>: Rule_Learning, <c1>: Neural_Networks, <c2>: Case_Based, <c3>: Genetic_Algorithms, <c4>: Theory, <c5>: Reinforcement_Learning, <c6>: Probabilistic_Methods]. Remember, your answer should be in the form of the class label.
<information>
<feature>
<center_node><x><x emb></x></center_node>
<1st_feature_similarity_graph><x><x emb></x></1st_feature_similarity_graph>
</feature>
</information>

[Assistant]: The answer is: <c6>

Note that the "<x emb>" is the text token embedding for feature feature "x" generated by the MLP projector discussed in Appendix C.1.

E Interactive Graph Reasoning

As shown in Figure 4, GraphText facilitates graph learning within a textual domain, it allows for direct interaction between both humans and AI agents. In this section, we spotlight the interactive graph reasoning capabilities of GraphText using a practical example. First, we demonstrate how

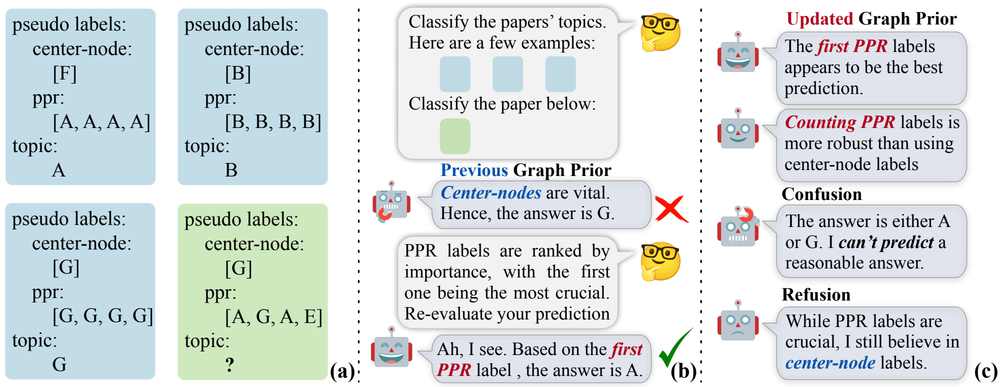


Figure 4: **(a)** Few-shot demonstrations (blue) and the target node #2188 (green) to predict on Cora. **(b)** An illustration of how human interaction changes the graph reasoning of an LLM, where the LLM previously has the prior that the center-node is vital. **(c)** Behaviors of LLMs after given demonstrations/human interaction: *update graph prior* to bias more on PPR (personalized pagerank); leads to *confusion* or *refusion*. Details are discussed in Section A.1.

GraphText can engage in self-interaction via zero-shot chain of thought reasoning. Following that, we illustrate how human interactions can guide GraphText to refine its graph reasoning approach.

E.1 Zero-shot Chain of Thought Reasoning

Below is the example of graph reasoning on Cora node #2188 in the setting of standard zero-shot chain of thought reasoning [41][10] The input prompt for "Cora node #2188" is as below:

Input Prompt Cora node #2188:
[Human]: You are a helpful assistant that generates a classifies the topic of an academic paper based on the labels of the cited papers. You are going to choose the correct answer from several choices of paper categories:[A: Theory, B: Reinforcement Learning, C: Genetic Algorithm, D: Neural Network, E: Probabilistic Method, F: Case Based, G: Rule Learning]
Here are a few examples:
Graph information:
pseudo labels:
    center-node:['F']
    ppr:['A', 'A', 'A', 'A']
Topic of paper: A

Graph information:
pseudo labels:
    center-node:['B']
    ppr:['B', 'B', 'B', 'B']
Topic of paper: B

Graph information:
pseudo labels:
    center-node:['C']
    ppr:['C', 'C', 'C', 'C']
Topic of paper: C

Graph information:

[10]ChatGPT and GPT-4 perform reasoning with explanation by default. Therefore, there is no necessity to add the additional "Let's think step-by-step" prompt for the text input.

pseudo labels:
    center-node:['D']
    ppr:['D', 'D', 'E', 'D']
Topic of paper: D

Graph information:
pseudo labels:
    center-node:['A']
    ppr:['E', 'E', 'A', 'A']
Topic of paper: E

Graph information:
pseudo labels:
    center-node:['F']
    ppr:['F', 'F', 'F', 'F']
Topic of paper: F

Graph information:
pseudo labels:
    center-node:['G']
    ppr:['G', 'G', 'G', 'G']
Topic of paper: G

Now let's answer the question below:
Graph information:
pseudo labels:
center-node:['G']
ppr:['A', 'G', 'A', 'E']
What's the topic of the paper given the information above? Valid choices are [A: Theory, B: Reinforcement Learning, C: Genetic Algorithm, D: Neural Network, E: Probabilistic Method, F: Case Based, G: Rule Learning]

As discussed in Section A.1, according to the demonstrations, the PPR pseudo-labels provide a more robust mechanism for label prediction. Utilizing either a count of PPR labels followed by a majority vote, or merely referencing the foremost PPR label, consistently results in the correct categorization in the given examples. Hence, based on these graph inductive biases derived from the demonstrations, we can reasonably figure out correct topic of the target paper should be A, which not only is the first entry, but also the predominant label in the PPR pseudo-label sequence. Now, let's use ChatGPT and GPT-4 to answer the problem. We performed 15 times of evaluation and here we show one sample of correct and incorrect responses below:

Correct Response without Human Interaction
[ChatGPT]: Among the PPR labels, 'A' appears twice, 'G' appears once, and 'E' appears once. 'A' is the most frequent label, so we classify the topic of the paper as 'A.'
Therefore, the topic of the paper is A: Theory.

[GPT-4]: Given the examples provided:
1. Whenever the center-node label is consistent with the majority of the PPR pseudo labels, the topic of the paper matches the center-node label.
2. However, when there's a discrepancy between the center-node label and the majority of the PPR labels, the topic of the paper is typically determined by the first (most crucial) label in the PPR list.
Given the graph information provided:
- The center node is labeled 'G' which corresponds to 'Rule Learning'.
- The 'ppr' nodes are labeled 'A', 'G', 'A', and 'E', which corresponds to 'Theory', 'Rule Learning', 'Theory', and 'Probabilistic Method' respectively.
It seems that while the center node is 'G', the cited papers are mostly of category 'A' and

’E’.
From the pattern observed in the examples, the category that has the majority or at least equal emphasis with other categories in the ’ppr’ nodes tends to be the category of the paper. Therefore, in this case, ’A’ or ’Theory’ has the highest count in the ’ppr’ nodes.
Thus, the topic of the paper given the information above is: A: Theory.

Wrong Response without Human Interaction
[ChatGPT]: Based on the labels in the ”ppr” list, we can see that ’A’ (Theory), ’G’ (Rule Learning), and ’E’ (Probabilistic Method) are mentioned. However, the ’center-node’ label is ’G’ (Rule Learning), which indicates a stronger association with ”Rule Learning.” So, the topic of the paper is most likely ”Rule Learning (G).”

[GPT-4]: Given the graph information: pseudo labels: center-node:[’G’] ppr:[’A’, ’G’, ’A’, ’E’]
The center node label is ’G’, which corresponds to ”Rule Learning”. The papers referenced (ppr) are mixed between ”Theory”, ”Rule Learning”, and ”Probabilistic Method”. However, since the center node holds the primary topic label, the academic paper’s topic is: G: Rule Learning.

From the analysis above, it’s clear that:

By recasting the graph learning challenge into problems that both humans and LLMs can interpret and reason with, GraphText facilitates LLMs in graph reasoning. For instance, ChatGPT can deduce the right answer through label counting, as highlighted in green. Impressively, GPT-4 discerns the correct solution by employing the Personalized PageRank (PPR) prior, as illustrated in blue, and identifies the first PPR neighbor as the most important factor for prediction.

However, the lucid explanations rendered in natural language allowed us to pinpoint certain inadequacies in their graph reasoning approach:

Failure to Retrieve Prior Knowledge: Table 2 shows that LLMs, particularly ChatGPT, don’t always effectively recall their previous understanding of graphs; only 26.7% of ChatGPT’s reasoning relies on PPR.

Persistence in Prior Belief: Despite the inherent reasoning prowess of LLMs, they can sometimes be hindered by preconceived notions about graphs. A detailed examination of the results in Table 2, accompanied by the explanations of reasoning above, indicates that a deep-seated bias - where the central node is viewed as the most vital (denoted in red) - is a primary cause of mistakes.

In the following section, we engage with *GraphText* to enhance its graph reasoning capabilities.

E.2 Graph Reasoning with Human Interaction

As highlighted in the preceding section, LLMs occasionally struggle with recalling prior knowledge and remain anchored to their pre-existing beliefs. To address this, we engage with GraphText by: (1) providing a clear definition of PPR, and (2) explicitly instructing them to assess the reliability of both center-node pseudo and PPR pseudo labels. This leads to the following message:

Human Interaction Note that PPR pseudo labels represent a sequence of labels ranked by importance, with the first one being the most crucial. Let’s evaluate the reliability of center node pseudo labels against PPR pseudo labels and then make another prediction.

With the interaction above, in most cases LLMs refined their answer, below are some examples:

Correct Reasoning after Human Interaction
[ChatGPT]: In this PPR list, the labels ’A’ and ’G’ are present, with ’A’ appearing before ’G,’ which means ’A’ is considered more crucial in the context of this paper. Now, let’s

make a prediction based on this evaluation: Since 'A' is the most crucial label in the PPR list, we will prioritize it for classifying the topic of the paper. Therefore, the paper's primary topic is "A: Theory" (Category A). The presence of 'G' in the PPR list indicates some relevance to "Rule Learning" (Category G), but because it comes after 'A,' it is considered less important in this context. So, the revised prediction is that the primary topic of the paper is "A: Theory."

[GPT-4]: Given the new information that PPR pseudo labels represent a sequence of labels ranked by importance, let's re-evaluate the provided data:
The center node suggests the paper is about 'G' (Rule Learning). However, considering the prominence of 'A' (Theory) in the ranked sequence, and since it's the most crucial label, the paper might be primarily focused on theoretical aspects related to Rule Learning. Considering both the center node and the importance sequence, the topic of the paper should be: A: Theory

Refusion after Human Interaction

[ChatGPT]: Certainly, let's evaluate the reliability of center node pseudo labels against PPR pseudo labels.
In the provided graph information: The center-node label is 'G,' which suggests that the most important label for this paper is 'G.' However, when we consider the PPR labels, we see that 'A' appears as the first and third label, which might indicate some importance as well.
Let's consider both the center-node and the PPR labels for our prediction. Given that 'G' is the center-node label and 'A' appears in the PPR labels, we can prioritize the center-node label but also consider the presence of 'A' in the PPR labels.
So, the prediction for the topic of the paper can be: Topic of paper: G (with some influence from A).

Confusion after Human Interaction

[ChatGPT]: Based on this information, we can classify the topic of the paper as a combination of the most frequent labels among the neighboring papers, which are 'A' and 'G'. Therefore, the topic of the paper is likely a combination of topics 'A' and 'G'.
So, the answer is a combination of two topics, and the closest match among the given options is:
Option: F: Case Based *# Note that F does not present in the pseudo labels of the question*

The provided examples, along with the consolidated findings in Table 2, compellingly show that both ChatGPT and GPT-4 can adjust their pre-existing biases about graphs when given human feedback. Notably, after this interaction, GPT-4 delivers a flawless accuracy rate of 100%, consistently following the PPR logic. Meanwhile, ChatGPT also sees a significant performance boost, with an accuracy improvement of 36.9%. However, as evidenced in the examples, ChatGPT occasionally refuses updating its predictions or becomes confused.

F Ethics Statement

Graphs are prevalent in the real world. On the bright side, GraphText alleviates the computational load and carbon footprint associated with training numerous non-transferable, graph-specific models. However, while the training-free graph reasoning capability of GraphText introduces minimal costs, there's potential for misuse in malicious recommendation systems and malware.

G Limitations and Future Work

While the GraphText framework offers notable advantages, there is ample room for enhancement and exploration of new applications.

One primary concern not fully addressed in this paper is the question of *how to discretize continuous features*. As evident from Table 6, most optimal settings are label-based. This makes GraphText resemble a label propagation model, with Citeseer being the only exception. We posit that the observed trend might be attributed to two main factors: 1) The inevitable information loss of the discretization process, and 2) The discord between feature and label spaces, making reasoning challenging for LLMs.

Additionally, the *design space of the graph-proxy tree is large* and often requires either expert knowledge or hyperparameter optimization. Although GraphText enjoys flexibility and effectiveness, crafting the text-feature set F and relation set R, and determining their combinations result in a vast search space. However, given that GraphText operates on a training-free paradigm, hyperparameter optimization can be swift.

Notwithstanding its constraints, GraphText introduces avenues for novel research. Chiefly, it sets the stage for graph reasoning in natural language. Emerging advancements in the LLM realm can potentially be integrated into the graph ML domain. This includes areas such as multi-step reasoning [45], decision-making [46, 25], tool utilization [36], and multi-agent collaboration [30, 19]. Furthermore, the prospect of training-free graph learning streamlines the validation process for graph model designs. If one assumes the optimal relation set R and feature set H to be transferable, significant training time can be saved. Researchers can quickly identify suitable settings with GraphText and then apply these configurations to the hyperparameters of other GNNs/LLMs.