LINKGPT: Teaching Large Language Models To Predict Missing Links

Zhongmou He, Jing Zhu, Shengyi Qian, Joyce Chai, Danai Koutra
University of Michigan

Abstract

Large Language Models (LLMs) have shown promising results on various language and vision tasks. Recently, there has been growing interest in applying LLMs to graph-based tasks, particularly on Text-Attributed Graphs (TAGs). However, most studies have focused on node classification, while the use of LLMs for link prediction (LP) remains understudied. In this work, we propose a new task on LLMs, where the objective is to leverage LLMs to predict missing links between nodes in a graph. This task evaluates an LLM's ability to reason over structured data and infer new facts based on learned patterns. This new task poses two key challenges: (1) How to effectively integrate pairwise structural information into the LLMs, which is known to be crucial for LP performance, and (2) how to solve the computational bottleneck when teaching LLMs to perform LP. To address these challenges, we propose LINKGPT, the first LLM-based training and inference framework specifically designed for LP tasks on homogeneous TAGs. To enhance the LLM's ability to understand the underlying structure, we design a two-stage instruction tuning approach where the first stage finetunes the pairwise encoder, projector, and node projector, and the second stage further finetunes the LLMs to predict links. To address the efficiency challenges at inference time, we introduce a retrieval-reranking scheme and investigate three LLM-based retrieval methods. Extensive experiments show that LINKGPT can achieve state-of-the-art performance on real-world graphs and superior generalization in zero-shot and few-shot learning, surpassing existing benchmarks. At inference time, it can achieve $10\times$ speedup while maintaining high LP accuracy.

1 Introduction

Graph-structured data is ubiquitous in real-world applications, ranging from social networks to recommendation systems. Among the various tasks performed on graphs, link prediction (LP) is of particular importance. In recent years, graph neural networks (GNNs) that learn node representations by aggregating information from their local neighborhoods have been widely used for LP [39, 41, 20]. However, GNNs struggle to capture long-range dependencies and complex semantic information present in text-attributed graphs (TAGs) and fail to generalize to unseen graphs [15].

Recent advancements in Large Language Models (LLMs) have revolutionized the natural language processing field [1, 19, 32, 21]. Inspired by their success, researchers have begun exploring the application of LLMs to graph-based tasks. However, most existing works focus on node classification. It remains unclear whether and how LLM can be used for LP, a task that relies heavily on structural information [31, 11]. Therefore, we propose to explore the ability of LLMs to perform link prediction tasks. Note that in this paper, we primarily focus on LP on homogeneous TAGs, *i.e.*, TAGs with only one type of nodes and edges.

Nonetheless, teaching LLMs to perform LP is not straightforward and presents two key challenges. First, LLMs are primarily designed to process textual and sequential data. The way to integrate

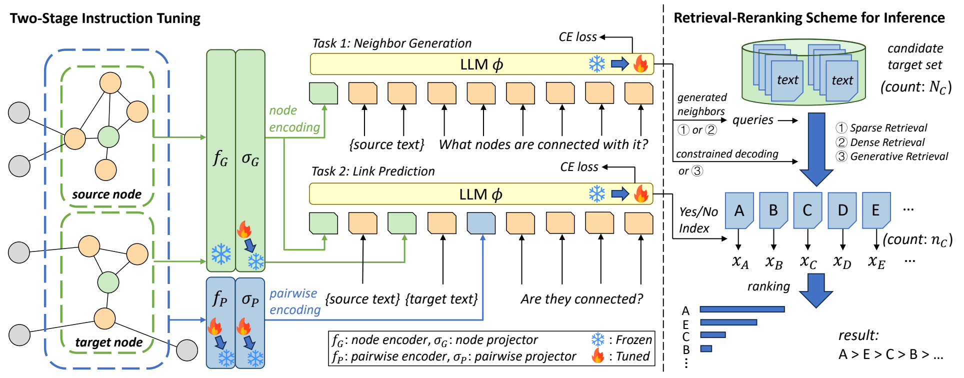


Figure 1: Overview of the LINKGPT framework. LINKGPT consists of two main components: (1) two-stage instruction tuning approach to incorporate structural information into LLMs, and (2) a retrieval-reranking scheme to address the computation bottleneck at inference time. Note that, 🔥 ⇒ ❄ means that this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

node-wise and pairwise structural information into LLMs is non-trivial. Second, teaching LLMs to predict missing links presents computational bottlenecks. LP typically requires ranking a large number of candidate target nodes (*e.g.*, 1,000) for each source node at inference time, which is computationally expensive when combined with LLMs.

To address these challenges, we propose LINKGPT, the first LLM-based training and inference framework specifically designed for link prediction on homogeneous TAGs. LINKGPT consists of two main components: (1) a two-stage instruction tuning approach to incorporate neighborhood and pairwise structural information into LLMs, and (2) a retrieval-reranking scheme to address the computation bottleneck at inference time. Specifically, during inference, we first retrieve a smaller subset of candidate target nodes by generating potential neighbors of the source node. Then, we evaluate each candidate based on their textual, neighborhood, and pairwise information using the LLM. Extensive experiments show that LINKGPT not only achieves state-of-the-art performance across datasets but also possesses strong generalization ability. Moreover, by employing our designed retrieval-reranking scheme for inference, LINKGPT is $10\times$ faster while maintaining high LP accuracy.

Details about related works are presented in Appendix A.

2 Preliminaries

Graphs. Formally, a **graph** is denoted by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{X}\}$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of edges, and $\mathcal{X}$ represents the set of attributes or features associated with each node in $\mathcal{V}$. A graph $\mathcal{G}$ is considered to be a **text-attributed graph (TAG)** if its node attributes $\mathcal{X}$ are texts. In this paper, we mainly focus on homogeneous TAGs, *i.e.*, TAGs with only one type of nodes and edges. We further let $\mathcal{N}_u^k$ denote the k**-hop neighbors** of node u, which is the set of nodes at a distance of k from u.

Link prediction. Given a source node $s \in \mathcal{V}$ and a set of candidate target nodes $\mathcal{C} = \{t_1, t_2, \cdots, t_{N_C}\} \subseteq \mathcal{V}$, which consists of 1 positive target node and $N_C - 1$ negative target node, the link prediction task aims to rank all candidate target nodes based on the probability that there is a link between the source node s and each candidate target node t_i. The performance is evaluated by how highly the positive candidate node is ranked among all candidates.

3 LINKGPT

In this section, we present the proposed LINKGPT framework for link prediction on text-attributed graphs (TAGs). LINKGPT leverages the power of LLMs to effectively capture both graph structure and textual semantics, enabling accurate and efficient link prediction. An overview of the LINKGPT is illustrated in Figure 1.

3.1 Node and Pairwise Encoding

To enable the LLM to effectively capture the graph structural, textual, and pairwise semantics, we introduce two special tokens: `<NODE>` and `<PAIRWISE>`. The `<NODE>` token encodes information about a node and its surrounding neighborhood, while the `<PAIRWISE>` token encodes the pairwise relationship between two nodes.

Neighborhood-aware node encoding. In a graph, the neighborhood $\mathcal{N}_v^1$ of node v plays a crucial role in understanding v itself [15]. Inspired by [31, 21], we obtain neighborhood-aware node embeddings through contrastive graph-text pre-training so that the node encoding contains the textual information of the node itself and its neighbors. We first obtain the neighborhood text representations $\mathbf{T} \in \mathbb{R}^{N \times 2d_T}$ and the node representations $\mathbf{H} \in \mathbb{R}^{N \times d_H}$ through BERT [8] and GraphFormers [37], separately (details are provided in Appendix B). The contrastive loss $\mathcal{L}_G$ is calculated as follows:

$$\Gamma = \text{norm}(\mathbf{H})\text{norm}(\mathbf{T})^T/\tau, \quad \mathcal{L}_G = \left((\text{CE}(\Gamma, \mathbf{y}) + \text{CE}(\Gamma^T, \mathbf{y})\right)/2 \tag{1}$$

where τ denotes the temperature, and the label $\mathbf{y}$ is set to $(0, 1, \cdots, B-1)^T$, where B denotes the batch size. The learned node encoding $\mathbf{H}$ is then mapped to the semantic space of the LLM using a simple alignment projector σ_G and will be used the embedding of the special token `<NODE>`.

Pairwise encoding. To capture the pairwise relationship between nodes, we employ LPFormer [29]. For node a and b, the LPFormer learns a pairwise encoding $f_P(a, b)$ to represent their relationship.

$$f_P(a, b) = \sum\nolimits_{u \in \mathcal{V}_{(a,b)}} w(a, b, u) \odot h(a, b, u) \tag{2}$$

where $w(a, b, u)$ and $h(a, b, u)$ denote the importance and encoding of node u relative to the relation between a and b respectively, and $\mathcal{V}_{(a,b)}$ denotes the set of nodes that might be important to the relation. $f_P(a, b)$ is also mapped to the semantic space of LLM using an alignment projector σ_P. Note that the pairwise encoder is not pre-trained. Details about pairwise encoder are in Appendix C.

3.2 Two-Stage Instruction Tuning

We leverage a two-stage instruction tuning approach to train LinkGPT for link prediction and neighbor generation tasks. The prompts for these tasks are shown in Appendix D.

Training schedule. We adopt a two-stage tuning strategy. In stage 1, we focus on training the pairwise encoder and aligning the two special encodings with the word embeddings of the pre-trained LLM. The LLM is kept frozen, and only the encoding-related modules, including the pairwise encoder and two alignment projectors, are tuned. In stage 2, we keep the encoding-related modules frozen and tune the LLM through LoRA [12] to improve its understanding of the encodings.

3.3 Retrieval-Reranking Scheme for Inference

The inference stage of link prediction involves ranking a set of candidate target nodes $\mathcal{C} = \{t_1, t_2, \cdots, t_{N_C}\}$ for each source node s. However, directly ranking all candidates using the LLM can be computationally expensive, especially when the candidate set $\mathcal{C}$ is large. To address such a challenge, we propose a retrieval-reranking scheme that efficiently shrinks the size of the candidate set before applying the LLM for final ranking.

Retrieval stage. In the retrieval stage, our goal is to quickly identify a smaller subset of $n_C \ll N_C$ candidate target nodes that are most likely to be connected to the source node. We first prompt the LLM to generate potential neighbors using the neighbor generation prompt in Figure 3, and then retrieve n_C candidates from $\mathcal{C}$. We explore three effective retrieval methods, sparse retrieval (SR), dense retrieval (DR), and generative retrieval (GR). SR and DR use BM25 [26] and the cosine similarity of the embeddings as the scoring functions, separately, while GR directly "generates" most possible candidates in $\mathcal{C}$. Details about the retrieval methods are in Appendix E.

Furthermore, to utilize pairwise heuristics, we apply a distance-based grouping strategy for all three retrieval methods, where we select βn_C nodes from the source node's 2-hop neighborhood $\mathcal{N}_s^2$ and $(1-\beta)n_C$ nodes from the rest of the graph $\mathcal{V} \backslash \mathcal{N}_s^2$ with the highest scores. Here we set $\beta = 0.65$.

Reranking stage. In the reranking stage, we use the LLM to rank the retrieved n_C candidates and obtain the final link prediction results. To quantify the probability of an edge existing between the source node s and a candidate target node t_i, we use the ratio of the probability of the LLM outputting "Yes" to the probability of the LLM outputting either "Yes" or "No" as the final score.

4 Experiments

Table 1: Performance of LinkGPT and baseline models on the link prediction task on 7 datasets. The best results are highlighted with a grey background, while the second-best results are underlined.

Dataset	Sports		Clothing		Math		Geology		ogbn-Arxiv		PubMed		Cora	
	MRR	H@1	MRR	H@1	MRR	H@1	MRR	H@1	MRR	H@1	MRR	H@1	MRR	H@1
GCN [17]	70.44	60.17	68.18	60.00	51.35	40.11	45.80	33.14	69.19	58.6	34.96	23.04	28.92	16.01
GraphSAGE [10]	77.60	68.43	81.17	71.61	50.97	36.77	44.17	28.41	77.93	67.42	39.28	24.12	33.61	19.75
GATv2 [3]	81.44	72.96	87.83	81.56	65.65	55.27	51.59	37.95	78.62	68.14	39.33	26.19	47.50	33.24
SimKGC [34]	89.42	84.52	89.95	84.23	73.57	62.94	64.38	51.56	85.53	77.90	58.11	43.96	46.76	33.08
SEAL [40]	76.79	69.58	82.14	75.41	61.79	56.34	58.05	50.07	86.23	78.29	52.41	39.17	46.88	34.83
BUDDY [4]	81.33	74.00	83.89	76.52	58.15	48.30	54.95	45.32	82.45	75.48	40.68	29.53	34.26	22.85
LPFormer [29]	69.94	62.61	65.99	56.29	47.97	42.19	43.10	35.08	74.74	65.69	30.08	20.78	30.12	22.47
LLaMA2 [32]	40.81	30.88	30.22	22.49	22.92	13.54	21.86	13.47	46.53	33.81	37.98	24.47	19.06	11.99
GraphGPT [31]	14.82	5.96	32.29	14.28	12.43	4.37	9.78	2.62	9.78	2.20	7.49	1.82	5.34	1.57
LLaGA [5]	83.41	75.40	84.49	77.54	74.25	63.34	62.19	49.80	78.74	68.73	43.44	29.05	42.87	29.53
LinkGPT (Ours)	87.07	79.56	90.18	84.82	81.03	71.01	75.43	64.57	88.89	82.84	66.54	53.42	51.57	36.70

4.1 Experimental Setup

Datasets. Following previous work on LP and TAGs [5, 15], We train and evaluate our model on the following 7 datasets: Amazon-Sports, Amazon-Clothing [24], MAG-Geology, MAG-Math [30, 42], ogbn-Arxiv [13], PubMed, and Cora [28]. These datasets exhibit notable diversity in terms of size, sparsity, and domain. Detailed statistics about the 7 datasets are presented in Appendix G.

Baselines. We apply four categories of competitive baseline models, which include (1) traditional GAE-based models (GCN [17], GraphSAGE [10], and GATv2 [3]), (2) knowledge graph completion models (SimKGC [34]), (3) models that make use of nodes' pairwise information (SEAL [40], BUDDY [4], and LPFormer [29]), and (4) LLM-based models (vanilla LLaMA2 [32], GraphGPT [31], and LLaGA [5]. More descriptions of baseline models can be found in Appendix H.

Moreover, the implementation details of LinkGPT are in I.

4.2 Overall Performance

We first evaluate the overall performance of our LinkGPT framework and compare its performance with the state-of-the-art baselines on the link prediction task. The number of candidates N_C is set to be 150 for all datasets and the retrieval stage of LinkGPT is not applied for a clearer assessment. The results are presented in Table 1 and the experimental errors are reported in Appendix J. The proposed LinkGPT model significantly outperforms all the baseline models across all datasets except Sports on the link prediction task, achieving state-of-the-art performance in terms of both MRR and H@1. This highlights the effectiveness of LinkGPT's approach in leveraging node-wise, pairwise, and textual information for link prediction.

4.3 Generalization Ability

We then evaluate the generalization ability of LinkGPT. Note that in this section we still set $N_C = 150$ and do not apply the retrieval stage. We train LinkGPT on one dataset A and evaluate it on another dataset B, without fine-tuning or additional examples in the prompt. Results are summarized in Table 2. The proposed LinkGPT model demonstrates the strongest generalization ability, outperforming all other baseline models in all settings. LinkGPT not only benefits from its LLM backbone, which possesses powerful text feature generalization capabilities, but also from effectively capturing the structural commonalities of link formation through the node and pairwise encodings. We also investigate the few-shot generalization ability of LinkGPT in Appendix K.

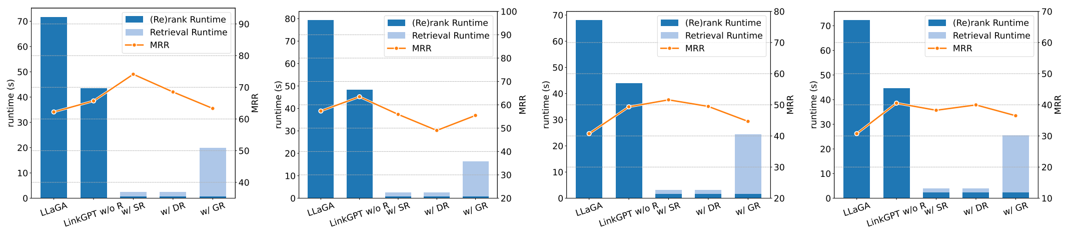
Table 2: Zero-shot performance on the link prediction task. $A \rightarrow B$ indicates that the model is trained on dataset A and evaluated on an unseen dataset B.

Dataset	Cross-category				Cross-domain			
	Sports→Clothing		Math→Geology		Sports→Math		Math→Sports	
	MRR	H@1	MRR	H@1	MRR	H@1	MRR	H@1
GraphSAGE [10]	51.88	36.43	8.11	2.06	19.27	8.88	27.64	14.52
GATv2 [3]	67.01	55.36	21.41	13.54	36.87	26.86	49.68	37.21
SimKGC [34]	65.58	54.20	46.68	35.78	58.09	47.72	65.77	53.65
BUDDY [4]	81.23	72.01	24.45	13.35	26.43	15.73	42.39	21.22
LPFormer [29]	68.26	59.64	42.70	35.08	48.50	43.75	56.78	44.13
LLaMA2 [32]	30.22	22.49	21.86	13.47	22.92	13.54	40.81	30.88
LLaGA [5]	78.62	68.90	23.20	13.13	23.86	12.97	53.67	40.46
LINKGPT (Ours)	85.74	79.01	71.58	60.42	67.50	55.27	69.38	55.40

4.4 Scalability Analysis

In this section, we investigate the scalability of LINKGPT by evaluating the performance and runtime of LINKGPT and another powerful LLM-based baseline, LLaGA [5], with large candidate target sets. The number of negative candidates N_C is set to 1,800 for all datasets. For the number of candidates to retrieve n_C, we set $n_C = 30$ for Sports and Clothing and $n_C = 60$ for all other datasets.

We conduct experiments in the following five settings: 1. LLaGA [5], 2. LINKGPT w/o Retrieval, 3. LINKGPT w/ SR, 4. LINKGPT w/ DR, and 5. LINKGPT w/ GR. As shown in Figure 2, thanks to LINKGPT's neighbor generation capability, including the retrieval stage can boost inference efficiency while maintaining good performance. For LINKGPT w/ SR, the runtime decreases by a factor of 10 but the MRR decline does not exceed 10%. Due to the page limitation, the results for ogbn-Arxiv, PubMed, and Cora are shown in Figure 6. We also include a case study in Appendix L.



(a) Sports (b) Clothing (c) Math (d) Geology

Figure 2: Performance of the retrieval-reranking scheme of LINKGPT on Sports, Cloth, Geology, and Math. The line chart represents the final MRR, using the right axis, while the bar chart represents the total runtime required for each $(s, \mathcal{C})$ pair during inference, using the left axis.

For the individual contribution of the node and pairwise encodings, one may refer to the ablation study in Appendix M.

5 Conclusion

In this paper, we introduced LINKGPT, a novel approach that leverages the power of Large Language Models (LLMs) for link prediction tasks on text-attributed graphs (TAGs). By combining node and pairwise encodings, two-stage finetuning, and a retrieval-reranking scheme for inference, LINKGPT effectively addresses the challenges of incorporating pairwise node information and efficiently ranking a large number of candidates. Our extensive experiments on real-world datasets demonstrate LINKGPT's superior performance and computational efficiency compared to state-of-the-art methods. Furthermore, the zero-shot and few-shot generalization experiments highlight LINKGPT's robustness and ability to transfer knowledge across different categories and domains.

References

[1] 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.

[2] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Networks*, 30:107–117, 1998.

[3] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *ArXiv*, abs/2105.14491, 2021.

[4] Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Hammerla, Michael M. Bronstein, and Max Hansmire. Graph neural networks for link prediction with subgraph sketching, 2023.

[5] Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. Llaga: Large language and graph assistant, 2024.

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[7] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL-HLT*, 2019.

[9] Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

[10] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Neural Information Processing Systems*, 2017.

[11] 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*, 2023.

[12] J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685, 2021.

[13] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *NeurIPS*, 33:22118–22133, 2020.

[14] Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. Large language models on graphs: A comprehensive survey. *arXiv preprint arXiv:2312.02783*, 2023.

[15] Bowen Jin, Wentao Zhang, Yu Zhang, Yu Meng, Xinyang Zhang, Qi Zhu, and Jiawei Han. Patton: Language model pretraining on text-rich networks. *arXiv preprint arXiv:2305.12268*, 2023.

[16] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18:39–43, 1953.

[17] Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907, 2016.

[18] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[19] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[20] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *CIKM*, pages 556–559, 2003.

[21] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023.

[22] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[23] Víctor Martínez, Fernando Berzal, and Juan-Carlos Cubero. A survey of link prediction in complex networks. *ACM computing surveys (CSUR)*, 49(4):1–33, 2016.

[24] Julian McAuley, Christopher Targett, Javen Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015.

[25] Mark E. J. Newman. Clustering and preferential attachment in growing networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 64 2 Pt 2:025102, 2001.

[26] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, apr 2009.

[27] Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. Sequence-to-sequence knowledge graph completion and question answering. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2814–2828, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[28] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Mag.*, 29(3):93–106, sep 2008.

[29] Harry Shomer, Yao Ma, Haitao Mao, Juanhui Li, Bo Wu, and Jiliang Tang. Lpformer: An adaptive graph transformer for link prediction. *arXiv preprint arXiv:2310.11009*, 2023.

[30] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, page 243–246, New York, NY, USA, 2015. Association for Computing Machinery.

[31] 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. *arXiv preprint arXiv:2310.13023*, 2023.

[32] 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.

[33] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. *ICLR*, 2020.

[34] Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. SimKGC: Simple contrastive knowledge graph completion with pre-trained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4281–4294, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[35] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks, 2020.

[36] Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Weihao Han, Hao Sun, et al. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. *Advances in Neural Information Processing Systems*, 36:17238–17264, 2023.

[37] Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. Graphformers: Gnn-nested transformers for representation learning on textual graph. In *Neural Information Processing Systems*, 2021.

[38] Liang Yao, Jiazhen Peng, Chengsheng Mao, and Yuan Luo. Exploring large language models for knowledge graph completion, 2024.

[39] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *NeurIPS*, 31, 2018.

[40] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Neural Information Processing Systems*, 2018.

[41] 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. *NeurIPS*, 34:9061–9073, 2021.

[42] Yu Zhang, Bowen Jin, Qi Zhu, Yu Meng, and Jiawei Han. The effect of metadata on scientific literature tagging: A cross-field cross-model study. In *WWW'23*, pages 1626–1637, 2023.

[43] 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. *arXiv preprint arXiv:2210.14709*, 2022.

[44] Jing Zhu, Xiang Song, Vassilis N Ioannidis, Danai Koutra, and Christos Faloutsos. Touchup-g: Improving feature representation through graph-centric finetuning. *arXiv preprint arXiv:2309.13885*, 2023.

[45] Yun Zhu, Yaoke Wang, Haizhou Shi, and Siliang Tang. Efficient tuning and inference for large language models on textual graphs. *arXiv preprint arXiv:2401.15569*, 2024.

[46] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. *NeurIPS*, 34:29476–29490, 2021.

A Related Works

Text-attributed graphs. In this work, we focus on text-attributed graphs, where each node is associated with a text description. Text-attributed graphs are typically dense and homogeneous and are structurally different from heterogeneous, sparse knowledge graphs [36, 46]. As a result, knowledge graph-based methods are not typically used in text-attributed graphs. Recent studies on text-attributed graphs focus on learning the integration of textual semantics within individual nodes and the topological connections across nodes [15, 36, 43, 44]. LINKGPT further extends the study of learning on text-attributed graphs and proposes to incorporate structural information directly into the LLMs.

Generation-based knowledge graph completion (KGC). In recent years, some studies have approached KGC as a sequence-to-sequence task, where generative language models directly generate tail entity texts based on a given head entity and relation [27, 38]. The probability of LLM generating these texts is then used as the basis for ranking. Although seems similar to the generation-based retrieval approaches in LINKGPT, these methods cannot be directly applied to LP on general TAGs. Firstly, compared to the entity texts in KGs, the texts associated with nodes in general TAGs are often longer and more irregular, making it challenging for LLMs to generate text that matches the nodes present in the candidate set. Moreover, using the probability of text generation as the ranking criterion leads to a training scenario with only positive samples and no negative samples, and LLMs thus cannot learn the differences between the pairwise and neighborhood information between positive and negative target candidates, which are crucial for LP on general dense homogeneous TAGs.

Link prediction. Link prediction aims to complete missing links in a graph [23, 33]. While heuristic algorithms were once predominant, graph neural networks (GNNs) have been prevalent in the past few years. Methods that use GNNs for LP mainly fall into two categories: Graph Autoencoder (GAE)-based methods and enclosing subgraph-based methods [18, 4, 39]. While GNN-based methods show promising results, they do not generalize to unseen graphs. In this work, we aim to explore LLMs' structure reasoning ability on LP tasks and evaluate their generalization ability in both zero-shot and few-shot settings.

Large language models for graphs. Recent progress on applying LLMs for TAGs aims to leverage the power of LLMs to boost performance on graph-related tasks [11, 14]. Two main strategies have emerged: (1) LLMs as predictors, where LLMs directly generate solutions to tasks [31, 5], and (2) LLMs as enhancers, which utilize LLM capabilities to improve the representations learned by smaller GNNs for better efficiency [45]. LINKGPT directly predicts missing links in graphs and thus falls into the first category. While previous works mainly focus on improving LLMs' ability to predict node labels on node classification task, here we believe that LLMs' ability to predict missing links in graphs presents its actual structure reasoning ability.

B Details about Node Encoding

Let f_G and f_T denote the node encoder and text encoder, respectively. The neighborhood text representations $\mathbf{T} \in \mathbb{R}^{N \times 2d_T}$ are obtained by:

$$\mathbf{T}' = f_T(\mathcal{X}), \quad \mathbf{T}_v = \text{concat}\left(\mathbf{T}'_v, \frac{1}{|\mathcal{N}_v^1|} \sum\nolimits_{u \in \mathcal{N}_v^1} \mathbf{T}'_u\right) \tag{3}$$

where $\mathbf{T}'_v \in \mathbb{R}^{d_T}$ is the raw text encoding of node v. Here we use `bert-base-uncased` [8] as f_T.

Furthermore, the node representations $\mathbf{H} \in \mathbb{R}^{N \times d_H}$ are given by $\mathbf{H} = f_G(\mathcal{V}, \mathcal{E}, \mathcal{X})$. We employ GraphFormers [37] as our node encoder f_G, where layerwise GNN components are nested alongside the transformer encoder blocks of language models. Formally, the forward pass of each layer in GraphFormers is given by:

$$z_v^{(l)} = \text{GNN}(\{\mathbf{H}_u^{(l)}[\text{CLS}] \mid u \in \mathcal{N}_v^1\}) \tag{4}$$

$$\tilde{\mathbf{H}}_v^{(l)} = \text{concat}(z_v^{(l)}, \mathbf{H}_v^{(l)}) \tag{5}$$

$$\tilde{\mathbf{H}}_v^{(l)'} = \text{LayerNorm}(\mathbf{H}_v^{(l)} + \text{MHA}_{\text{asy}}(\tilde{\mathbf{H}}_v^{(l)})) \tag{6}$$

$$\mathbf{H}_v^{(l+1)} = \text{LayerNorm}(\tilde{\mathbf{H}}_v^{(l)'} + \text{MLP}(\tilde{\mathbf{H}}_v^{(l)'})) \tag{7}$$

where $\mathbf{H}_v^{(l)}$ is the hidden states of node v in the l -th layer, and MHA_{asy} means asymmetric multihead attention. The initial embedding $\mathbf{H}_v^{(0)}$ is obtained through a learnable embedding module.

C Details about Pairwise Encoding

To capture the pairwise relationship between nodes in an explicit and adaptive way, we employ LPFormer [29] as the pairwise encoder. For nodes $a, b \in \mathcal{V}$, LPFormer learns a pairwise encoding $f_P(a, b)$ to represent their relationship. Common heuristics, such as CN [25] and Katz Index [16], can be represented as its special cases. Formally,

$$f_P(a, b) = \sum\nolimits_{u \in \mathcal{V}_{(a,b)}} w(a, b, u) \odot h(a, b, u) \tag{8}$$

Link Prediction:
This is the source node. `<NODE>` Text: *{text}*.
This is another node. `<NODE>` `<PAIRWISE>` Text: *{text}*. Is this node connected with the source node? Answer: *{Yes/No}*
This is another node. `<NODE>` `<PAIRWISE>` Text: *{text}*. Is this node connected with the source node? Answer: *{Yes/No}*
...
Neighbor Generation:
This is the source node. `<NODE>` Text: *{text}*. What nodes are connected with it?
Answer: Text: *{text}*. Text: *{text}*. ...

Figure 3: Instruction templates for the *link prediction* and *neighbor generation* tasks. Only the answers (highlighted in green) are used for calculating the loss.

where $w(a, b, u)$ and $h(a, b, u)$ denote the importance and encoding of node u relative to the relation between a and b respectively, and $\mathcal{V}_{(a,b)}$ denotes the set of nodes that *might* be important to the relation, consisting of nodes with Personalized PageRank (PPR) [2] score higher than a threshold η relative to either a or b. In this paper, we set $\eta = 0.01$ for nodes in $\mathcal{N}_a^1 \cup \mathcal{N}_b^1$, and $\eta = 1$ for all other nodes to filter them out. Furthermore, $w(a, b, u)$ and $h(a, b, u)$ are given by:

$$\tilde{w}(a, b, u) = \phi_{\text{attn}}(\mathbf{h}_a, \mathbf{h}_b, \mathbf{h}_u, \mathbf{rpe}_{(a,b,u)}) \tag{9}$$

$$w(a, b, u) = \frac{\exp(\tilde{w}(a, b, u))}{\sum_{v \in \mathcal{V} \setminus \{a,b\}} \exp(\tilde{w}(a, b, u))} \tag{10}$$

$$h(a, b, u) = \mathbf{W}_P \text{concat}(\mathbf{h}_u, \mathbf{rpe}_{(a,b,u)}) \tag{11}$$

where ϕ_{attn} denotes the GATv2 attention mechanism [3], $\mathbf{W}_P$ is a trainable weight matrix, and $\mathbf{rpe}_{(a,b,u)}$ denotes the relative positional encoding (RPE), which is given by:

$$\mathbf{rpe}_{(a,b,u)} = \text{MLP}(\text{ppr}(a, u), \text{ppr}(b, u)) \tag{12}$$

The pairwise encoding $f_P(a, b)$ is also mapped to the semantic space of the LLM using a simple alignment projector σ_{P}, and the mapped encoding will be the embedding of the special token `<PAIRWISE>`. However, unlike node encoder f_G, f_P is trained along with the LLM from scratch.

D Prompts

The prompts for the link prediction and neighbor generation tasks are shown in Figure 3.

In the prompts, each data point contains one source node s and multiple candidate target nodes, with an equal number of positive and negative ones. The positive candidates are sampled from $\mathcal{N}_s^1$, while the negative candidates are sampled from $\mathcal{V} \backslash \mathcal{N}_s^1$ randomly. Such setup allows the model to compare the current candidate with other candidates that appear before it in the context, enabling the model to have in-context learning capabilities during inference.

E Details about the Retrieval Methods

E.1 Sparse retrieval (SR).

We first prompt the LLM to directly generate several potential neighbors of the source node s using the neighbor generation prompt in Figure 3. The generated texts are then used as queries $\mathcal{Q} = \{q_1, q_2, \cdots, q_{n_q}\}$ to search for relevant candidates in the target set $\mathcal{C}$ using BM25 [26], which evaluates the similarity between queries and candidates based on term frequency and inverse document frequency (TF-IDF). Formally, the score of each candidate target node $t \in \mathcal{C}$ is given by:

$$\text{score}(t) = \frac{1}{n_q} \sum_{i=1}^{n_q} \text{BM25}(t, q_i) \tag{13}$$

$$\text{BM}(t, q_i) = \sum\nolimits_{j=1}^{\text{len}(q_i)} \text{IDF}(q_{ij}) \cdot \frac{f(q_{ij}, t) \cdot (k_1 + 1)}{f(q_{ij}, t) + k_1 \cdot \left(1 - b + b \cdot \frac{\text{len}(t)}{l_C}\right)} \tag{14}$$

where q_{ij} is the j-th term of query q_i, IDF(q_{ij}) is the inverse document frequency of term q_{ij}, $f(q_{ij}, t)$ is the term frequency of term q_{ij} in the text of candidate t, l_C is the average length of the texts of all candidate target nodes in $\mathcal{C}$, and k_1 and b are two adjustable parameters. The definition of IDF is further given by:

$$\text{IDF}(q_{ij}) = \log \frac{N_C - n(q_{ij}) + 0.5}{n(q_{ij}) + 0.5} \tag{15}$$

where $N_C = |\mathcal{C}|$, and $n(q_{ij})$ is the number of candidates that contain term q_{ij} in their corresponding texts.

E.2 Dense retrieval (DR).

In DR, queries $\mathcal{Q}$ are generated in the same way as SR, but the similarity between queries and candidates is evaluated using the cosine similarity between their embeddings. Formally, the score of each candidate t is given by:

$$\text{score}(t) = \frac{1}{n_q} \sum\nolimits_{i=1}^{n_q} \cos(\mathbf{q}_i, \mathbf{t}) = \frac{1}{n_q} \sum\nolimits_{i=1}^{n_q} \frac{\mathbf{q}_i \cdot \mathbf{t}}{|\mathbf{q}_i||\mathbf{t}|} \tag{16}$$

where $\mathbf{q}_i$ and $\mathbf{t}$ denote the embeddings of q_i and t. Pretrained language model `sup-simcse-bert-base-uncased` [9] is used here to create the text embeddings.

E.3 Generative retrieval (GR).

Unlike SR and DR where the text generated by the LLM are used as queries, LLM with GR can directly generate the texts of candidates in $\mathcal{C}$ in an end-to-end way, and thus does not require external indexing [7]. Specifically, we construct a prefix tree (a.k.a trie) [6] for $\mathcal{C}$ and then use it to guide the LLM to perform constrained beam search, ensuring that the generated sequences are always the prefix of some target candidates during the whole generation process. GR **implicitly** uses the candidates' probabilities of being generated by the LLM as the scoring function.

$$\text{score}(t) = P(t \mid s; \theta_\phi) = \prod\nolimits_{i=1}^{\text{len}(t)} P(t_i \mid t_{<i}, s; \theta_\phi) \tag{17}$$

where t_i denotes the i-th token in the text of candidate t, $t_{<i}$ represents the sequence of tokens $t_1, t_2, \cdots, t_{i-1}$, and θ_ϕ represents the parameters of the LLM ϕ.

F Time Complexity Analysis of Reusing Keys and Values During Inference

Denote the length of the part about the source node and all examples in the prompt by m_s, and denote the length of the part about the candidate target node by m_t. Since the time complexity of each self-attention layer in LLM is $O(m^2)$, where m represents the length of the input sequence, if we do not reuse any past keys or values, the time complexity for all n_C instructions would be

$$O(n_C(m_s + m_t)^2) = O(n_C m_s^2 + n_C m_s m_t + n_C m_t^2) \tag{18}$$

However, for a given $(s, \mathcal{C})$ pair, the prompt for each candidate $t_i \in \mathcal{C}$ shares the same part about the source node s and examples. Thus, by reusing the keys and values generated by each self-attention layer of the shared part, we can reduce the time complexity to

$$O\left(m_s^2 + n_C \sum_{i=1}^{m_t} (m_s + i)\right) = O(m_s^2 + n_C m_s m_t + n_C m_t^2) \tag{19}$$

Due to the existence of examples and question text, m_s is usually much longer than m_t. Therefore, reusing keys and values can boost the efficiency of the ranking stage of LinkGPT without any performance loss.

G Statistics of the Datasets

The statistics of the 7 datasets are presented in Table 3. Since the experiments on ogbn-Arxiv can already demonstrate the scalability of our model on large graphs, Sports, Clothing, Math, and Geology are subsampled by randomly selecting 20,000 nodes.

For all datasets, we randomly split the edges in the graphs into training, validation, and test sets in a 9:0.5:0.5 ratio. For the validation and test sets, the negative target candidates corresponding to each source node s are randomly selected from $\mathcal{V} \backslash \mathcal{N}_s^1$. During both training and testing, all edges in the test set are removed from the original graph to prevent data leakage.

Table 3: Datasets used in LINKGPT: We use 7 datasets of various scales and domains and are commonly used for LP on TAGs [15, 5].

	Sports [15]	Clothing [15]	Math [15]	Geology [15]	ogbn-Arxiv [5]	PubMed [5]	Cora [5]
Domain	e-commerce	e-commerce	academic	academic	academic	academic	academic
# Nodes	20,417	20,180	19,878	20,530	169,343	19,717	2,708
# Edges	48,486	44,775	34,676	51,540	1,166,243	44,338	5,429
Sparsity	2.33e-4	2.20e-4	1.76e-4	2.45e-4	8.13e-5	2.28e-4	1.48e-3

H Descriptions of the Baseline Models

In this section, we introduce the ten baseline models used in our study, which are organized into four categories.

For all GAE-based models, we use the implementation of Deep Graph Library (DGL) [35]. After obtaining the embedding of each node through the model, a 3-layer MLP is applied to determine whether two nodes are connected. For all other baselines, we use their official code for implementation, and the corresponding optimal hyper-parameters reported in the original papers are used for training. Specifically, for the experiments of LLaGA [5] on Cora, PubMed, and ogbn-Arxiv, as well as the experiments of GraphGPT [31] on ogbn-Arxiv and PubMed, we directly used the checkpoints released by the original authors because they had already been trained and tested on these datasets.

(1) *GAE-Based Models*

- **GCN** [17]: GCN extends CNN to graph-structured data, learning node features through information passing and aggregation between nodes and their neighbors.
- **GraphSAGE** [10]: GraphSAGE learns graph representations by sampling and aggregating features from a node's neighbors, which makes it particularly suitable for large graphs.
- **GATv2** [3]: GATv2 uses a dynamic attention mechanism to assign different weights to each node in the graph, thus capturing important relationships between nodes.

(2) *Models Leveraging Contrastive Learning*

- **SimKGC** [34]: SimKGC is primarily designed for knowledge graph completion. It conducts effective contrastive learning by collecting a large number of diverse negatives and applying a hardness-aware InfoNCE loss.

(3) *Models Leveraging Pairwise Information*

- **SEAL** [40]: SEAL encodes local subgraphs between node pairs using GNNs, thus capturing richer structural information.
- **BUDDY** [4]: BUDDY uses subgraph sketches as message passing without explicit subgraph construction and applies feature precomputation technique, making it scalable in both time and space.
- **LPFormer** [29]: LPFormer designs an adaptive pairwise encoding using the attention mechanism of GATv2, thus effectively capturing pairwise information between nodes.

(4) *LLM-Based Models*

Table 4: Link prediction performance of LinkGPT on Clothing and Geology datasets with standard deviation.

Dataset	MRR	H@1
Clothing	90.18 ± 0.37	84.82 ± 0.59
Geology	75.43 ± 0.22	64.57 ± 0.45

Table 5: Summary of parameters and memory usage for each stage of instruction tuning.

	# Total Params	# Params Tuned	Memory
Stage 1	6.7 B	5.26 M $\sim$ 5.85 M	27 GB
Stage 2	6.7 B	8.39 M	27 GB

- **LLaMA2** [32]: The LLaMA2 model is a large autoregressive language model trained on a large corpus. We use the LLaMA2-7B version in this paper.
- **GraphGPT** [31]: GraphGPT performs text-graph grounding through contrastive learning and uses graph matching, node classification, and link prediction as training tasks for dual-stage instruction tuning. Although it can perform link prediction, it is primarily designed for node classification.
- **LLaGA** [5]: LLaGA enables LLMs to understand graph structural information by reorganizing nodes into two kinds of structure-aware sequence. It is trained using three tasks: node description, node classification, and link prediction.

I Implementation Details of LinkGPT

In each stage of the instruction tuning, the *link prediction* and *neighbor generation* tasks are carried out for 1 epoch respectively for all datasets except Cora. Due to its small size, Cora undergoes 3 epochs for each task. In stage 1 of instruction tuning, only the pairwise encoder f_P, projector σ_P and the node projector σ_G are tuned. In stage 2, the aforementioned modules are frozen and only the LLM is tuned. When tuning the LLM, we apply LoRA [12] to the query and value projection modules with rank $r = 16$ and alpha $\alpha = 32$. During the whole training process, the learning rate is set to 3×10^{-4} and the batch size is set to 8. Furthermore, we use AdamW [22] as the optimizer.

One NVIDIA A40 GPU is used for training and inference, except for generative retrieval (GR), which requires 2 GPUs. The numbers of tuned parameters, and memory usage for each stage of the instruction tuning are summarized in Table 5. The training time is 1.27 hours for Cora, 66 hours for ogbn-Arxiv, and approximately 5 hours for Sports, Clothing, Math, Geology, and PubMed.

J Experimental Error

Given the time-consuming nature of training and inference with LLMs, following convention [5], we do not calculate errors for all experimental data. Instead, we replicate the experiments from Table 1 five times using two moderately sized datasets, Clothing and Geology. The results, along with the standard deviations, are reported in Table 4.

Source Node: An experimental study of the kinetics of decompression-induced crystallization in silicic melt.
Positive Target Candidate: Preeruption conditions and timing of dacite-andesite *magma* mixing in the 2.2 ka *eruption* at Mount Rainier.
Generated Neighbors:
1. Experimental study of the kinetics of decompression-induced crystallization in silicic *magma*.
2. Thermodynamics of decompression-induced crystallization in silicic *magma*.
3. Hydrothermal alteration of the 1980 Mt. St. Helens *eruption*.
4. Crystallization kinetics of silicic melt at 1200–1400 °C and 10–15 kbar: Implications for *magma* chamber dynamics.
5. Decompression-induced crystallization in silicic melt.

Figure 5: One example of the neighbor generation task on the Geology dataset. The five texts in the "Generated Neighbors" are the top-scoring results from the five beam groups in the setting of SR or DR. The words that appear both in the positive target node and the generated neighbors are marked with the same colors.

K Few-shot Generalization Ability of LinkGPT

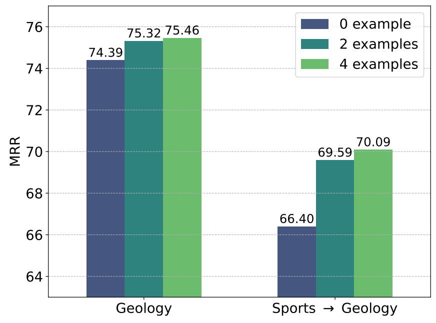


Figure 4: Performance of few-shot in-context learning. **(Left):** When the model is trained and evaluated on the same dataset, *e.g.*, Geology, we do not observe significant performance improvement for in-context learning. **(Right):** However, when we perform cross-domain generalization, *e.g.*, train on Sports and evaluate on Geology, we observe that in-context learning boosts the performance significantly.

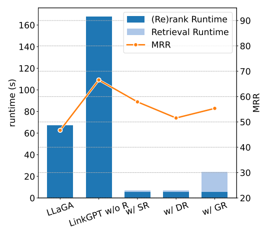
In this section, we evaluate the generalization ability of LinkGPT with the help of a few examples included in the prompt in test time. Note that none of the baseline models have the capability for in-context learning due to their architectures or prompt designs. The results are shown in Figure 4. When training and evaluating on the same dataset, in-context learning offers limited benefits since the model has already been finetuned on that dataset. However, for cross-domain generalization, in-context learning is particularly effective, increasing the model's MRR from 66.40 to 70.09 when trained on Sports and evaluated on Geology.

L Case Study of the Retrieval-reranking Scheme

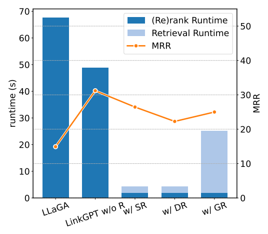
Figure 5 illustrates an example from Geology, where the five sentences in the "Generated Neighbors" part are the top-scoring results from the five beam groups in the setting of SR or DR. Although there is no perfect match, LinkGPT successfully predicts two crucial keywords: "magma" and "eruption." These two words did not appear in the source node or its neighbors but were mentioned multiple times in the generated texts, demonstrating the effectiveness of LinkGPT in neighbor generation.

M Ablation Study

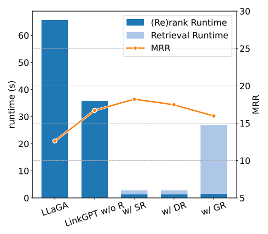
Setup. In this section, we investigate the individual contributions of the node and pairwise encodings. We remove each encoding component separately and train the model using the same two-stage instruction tuning strategy. This allows us to isolate the effect of each encoding on the model's



(a) ogbn-Arxiv



(b) PubMed



(c) Cora

Figure 6: Performance of the retrieval-reranking scheme of LINKGPT on ogn-Arxiv, PubMed, and Cora.

Table 6: Ablation on the node encoding and pairwise encoding. The pairwise encoding is more important than node encoding for the link prediction task since pairwise information is more helpful. However, both encodings contribute to the final performance of LINKGPT.

Dataset	Clothing		Geology	
	MRR	H@1	MRR	H@1
LINKGPT (Ours)	90.18	84.82	75.43	64.57
- w/o node encoding	88.77	82.46	74.73	63.50
- w/o pairwise encoding	83.08	74.11	72.07	60.42

performance. For the ablated models, we maintain the same hyper-parameters and training settings as the full LINKGPT model to ensure a fair comparison. The models are evaluated on the same benchmark datasets, and their link prediction performance is reported.

Results. The results of the ablation study are presented in Table 6. Incorporating the node and pairwise encodings leads to significant improvements in link prediction performance. This highlights the importance of explicitly encoding node and pairwise information to enhance the LLMs' understanding of the graph structure. The node encoding allows the model to capture node-wise structural information, such as the features of its neighborhood, and the pairwise encoding enables the model to reason about the relationships between node pairs.

Comparing the contributions of the two encodings, we find that the pairwise encoding plays a more crucial role in the link prediction task. The removal of pairwise encoding results in a larger performance drop compared to removing the node encoding. This observation aligns with the intuition that pairwise information, which captures the connectivity patterns between nodes, is more directly relevant to predicting missing links.