LOST-IN-DISTANCE: IMPACT OF CONTEXTUAL PROXIMITY ON LLM PERFORMANCE IN GRAPH TASKS

Hamed Firooz, Maziar Sanjabi, Wenlong Jiang*, & Xiaoling Zhai
Foundation AI Technologies (FAIT)
LinkedIn
{hfirooz, maz, wenjiang, xzhai}@linkedin.com

ABSTRACT

Large Language Models (LLMs) exhibit blind spots that impair their ability to retrieve and process relevant contextual data effectively. We demonstrate that LLM performance in graph tasks with complexities beyond the "needle-in-a-haystack" scenario—where solving the problem requires cross-referencing and reasoning across multiple subproblems *jointly*—is influenced by the proximity of relevant information within the context, a phenomenon we term "lost-in-distance". We examine two fundamental graph tasks: identifying common connections between two nodes and assessing similarity among three nodes, and show that the model's performance in these tasks significantly depends on the relative positioning of common edges. We evaluate three publicly available LLMs using various graph encoding techniques that represent graph structures for LLM input. Results indicate that model accuracy can decline by up to 6x as the distance between node connections increases, independent of graph encoding and model size.

1 INTRODUCTION

Large Language Models (LLMs) have achieved remarkable versatility by leveraging scale and attention-based architectures (Kaplan et al., 2020; Vaswani, 2017). They exhibit superhuman capabilities across diverse tasks such as language translation, reading comprehension, and question answering (Costa-jussà et al., 2022; Sanh et al., 2021), and serve as critical building blocks for applications beyond traditional language processing, including recommendation systems (Geng et al., 2022), graph-related tasks (Wang et al., 2024), and knowledge bases (AlKhamissi et al., 2022; Petroni et al., 2019). In particular, LLMs have shown potential in handling graph-based problems by encoding graph structures into textual formats, enabling general pre-trained models to address out-of-domain tasks (Sanford et al., 2024; Perozzi et al., 2024; Agarwal et al., 2020). Fine-tuned models like GraphWiz and GraphLM have demonstrated improved performance in graph reasoning tasks (Chen et al., 2024; Luo et al., 2024), emphasizing the promise of LLMs in this domain.

However, significant challenges remain in how LLMs process information in prompts, particularly for graph-related tasks. Research highlights issues such as the lost-in-the-middle phenomenon, where information placed in the middle of prompts is less effectively utilized (Liu et al., 2023), and performance degradation when contextual information is dispersed (Kaddour et al., 2023; Ge et al., 2024). In this work, we show that for graph tasks—particularly identifying common connections and measuring similarity—performance declines not only due to the absolute positioning of information but also because of the relative distance between relevant pieces, a limitation we term *lost-in-distance*. This phenomenon affects tasks requiring cross-subgraph reasoning, which is critical in applications such as molecular design, social network analysis, and recommendation systems (Tan et al., 2023; Xia et al., 2023; Gao et al., 2024).

Our analysis demonstrates that these limitations are universal across various LLMs, including Llama-3-8B, Llama-3-70B (Dubey et al., 2024), and GPT-4 (Achiam et al., 2023), and are compounded when "lost-in-the-middle" and "lost-in-distance" effects coexist. These findings underscore the need to address both the absolute and relative positioning of information to enhance LLM

*work is done while at LinkedIn FAIT

performance in graph-related tasks, which has profound implications for applying these models in domains requiring complex reasoning over structured data.

1.1 Notations and Definitions

We define a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ represent the sets of nodes and edges, respectively. If nodes v_i and v_j are directly connected, we denote the edge between them as $e_{ij} \in \mathcal{E}$. The neighbors of node v_i are defined as $\mathcal{N}(v_i) = \{v_k \in \mathcal{V} \mid e_{ik} \in \mathcal{E}\}$. A subgraph associated with node v_i is defined as $\mathcal{G}_{v_i} = (\{v_i\} \cup \mathcal{N}(v_i), \mathcal{E}_{v_i})$, where $\mathcal{E}_{v_i} = \{e_{ij} \mid e_{ij} \in \mathcal{E},\ v_j \in \mathcal{N}(v_i)\}$.

We use accuracy as the proportion of correct predictions to the total number of predictions made. More detail about notations can be found in Appendix A.

2 Graph Encoding and Graph Tasks

Representing graph-structured data as text is crucial for enabling LLMs to understand and answer questions about graphs, as different encodings can affect performance (Agarwal et al., 2020; Fatemi et al., 2024; Zhang et al., 2024a). In this work, we encode nodes as unique integers $v_i \in 0, 1, \ldots, n-1$ and experiment with three edge encoding functions from Fatemi et al. (2024): **Incident** (natural language adjacency lists), **Adjacency** (encoded as (v_i, v_j)), and **Expert** (encoded as $v_i \rightarrow v_j$). Since our graph tasks only require the subgraph and its structure, we include edge information only for the nodes of interest, a common practice in subgraph extraction (Shao et al., 2013). Figure 6 illustrates an example of these three encoding approaches in the prompt, where node 0 and node 1 are the nodes of interest.

We aim to analyze the performance of LLMs in two fundamental graph problems:

1. **Common Connection:** For this task, we ask LLMs to find the number of common connections between node v_i and node v_j, denoted as $|\mathcal{N}(v_i) \cap \mathcal{N}(v_j)|$.
2. **Similarity:** Given three nodes v_i, v_j and v_k sampled from a graph $\mathcal{G}$, we let v_j be the source node and v_i and v_k be the target nodes. The task for LLMs is to compare the number of common connections $|\mathcal{N}(v_i) \cap \mathcal{N}(v_j)|$ and $|\mathcal{N}(v_j) \cap \mathcal{N}(v_k)|$.

3 Experimentation

3.1 Experimental Setup

In this paper, we build upon previous studies (Huang et al., 2022; Fatemi et al., 2024; Zhang et al., 2024b) by conducting experiments on randomly generated graphs. We utilize the Erdős–Rényi (ER) graph generator (Erdős & Rényi, 1959) to create undirected graphs. We experiment with relatively large graphs comprising $n = 1000$ nodes. The undirected edge e_{ij} between nodes v_i and v_j is generated with probability $P(e_{ij} \in \mathcal{E})$. We set $P(e_{ij} \in \mathcal{E}) = 0.1$ throughout the main manuscript, and results for other values of $P(e_{ij} \in \mathcal{E})$ are presented in the Appendix for brevity.

Leveraging in-context learning (Dong et al., 2022; Wei et al., 2023), we conducted experiments using both closed-source models (GPT-4) and open-source models (Llama-3-8B-Instruct and Llama-3-70B-Instruct). For all models, we set the decoding temperature to zero to ensure the generation of deterministic answers. In each sample, we randomly selected two or three nodes as the nodes of interest for the common connection and similarity tasks, respectively. We conducted experiments on one hundred thousand randomly generated graphs to derive statistically significant insights into LLM behavior. The experimental results were then averaged across multiple samples.

3.2 Common Connection Task

In this section, we demonstrate the effect of increased distance on solving the common connection task. To create an input prompt for this task and to control the relative distance of relevant information (common neighbors), we use the following structured methodology: we sample two nodes from a graph, extract their subgraphs, group the common connections within each subgraph, and position

these connections at the beginning, middle, or end of their textual encoding. This recipe, specifically the grouping of relevant information into three positions—beginning, middle, and end (as illustrated in Figure 1 for adjacency encoding)—enables us to control the relative distance between common connections within the prompt. This allows us to investigate the effects on the model's performance when the relative distance is small, medium, or large. We denote the positions of relevant information within the first and second subgraphs as (p_1, p_2), where $p_1 \in \{0, 1, 2\}$ and $p_2 \in \{3, 4, 5\}$, respectively, for the sake of brevity.

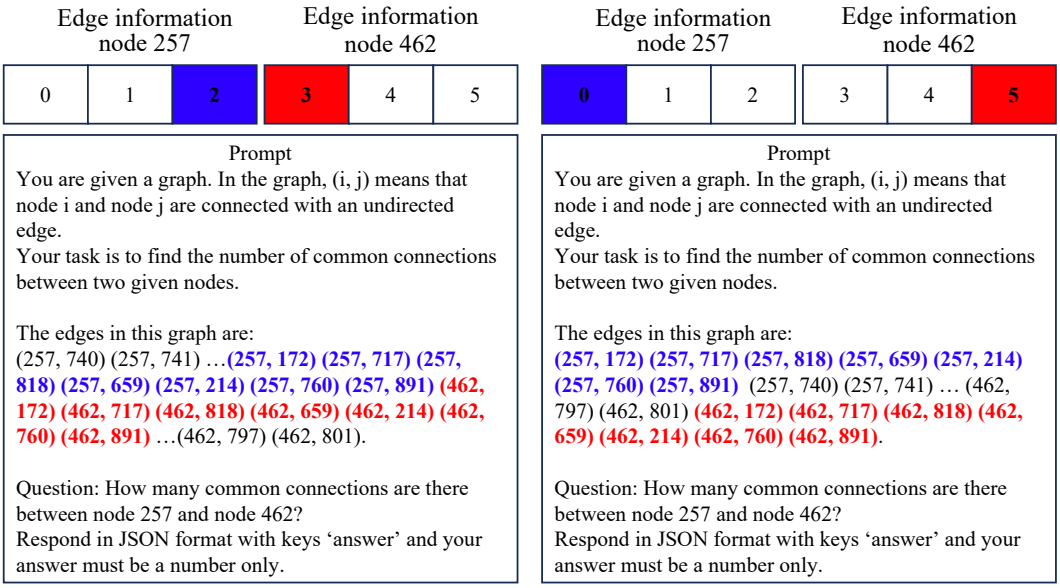

Figure 1: An example illustrating the placement of relevant information, highlighted in blue and red, at different positions using the adjacency encoding function for the common connection task.

The results in Figure 2 demonstrate that the model's performance in the common connection task is influenced by both the lost-in-the-middle phenomenon and the relative distance between common connections. When the position of relevant information is fixed in one subgraph, performance degrades as the other subgraph is positioned closer to the middle of the prompt, as seen in adjacency encodings where performance drops from 40% to 20% when shifting the second node's common connection from position 5 (end) to 3 (middle). However, across all three graph encodings, the model achieves optimal performance when relevant information is centrally located with minimal distance between components, particularly at positions $(2, 3)$, highlighting the interplay between lost-in-the-middle and lost-in-distance effects.

3.3 Similarity Task

Solving the similarity task for three nodes v_i, v_j, and v_k requires two common connection computations, $|\mathcal{N}(v_i) \cap \mathcal{N}(v_j)|$ and $|\mathcal{N}(v_j) \cap \mathcal{N}(v_k)|$, followed by a comparison, which inherently suffers from the lost-in-distance phenomenon. To measure this effect, we randomly shuffle edges within the subgraphs of the three nodes, position the source node v_j's subgraph ($\mathcal{G}_{v_j}$) at the center of the prompt, and place the other two subgraphs before and after it, mitigating the lost-in-the-middle effect. The median token distance between the common connections of the subgraphs quantifies the lost-in-distance effect. For more detail about the prompt please see appendix.

For brevity, we present the results of one encoding for each model in Figure 3, with all results summarized in Appendix G.2. Our findings indicate that when both distances are minimal, $(small, small)$ distance, GPT-4 and Llama-3-70B-Instruct exhibit the best performance. Llama-3-8B-Instruct, which has a high failure rate in following instructions as described in Appendix E.2, demonstrates the second-best performance, though it is not significantly different from the top performers.

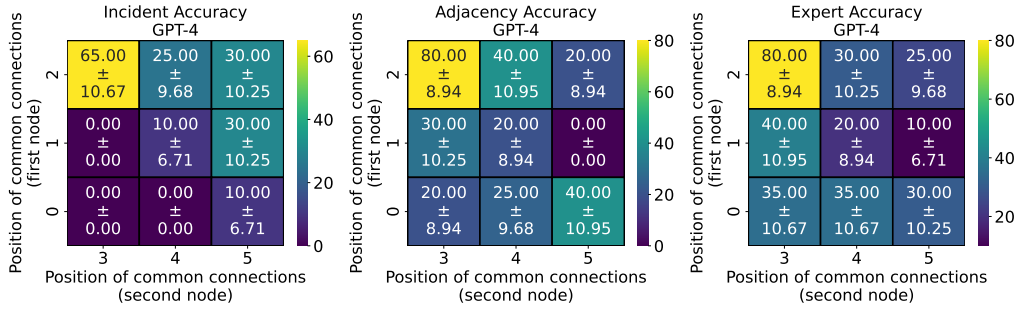

Figure 2: The effect of lost-in-distance on the common connection task. The number in each block is accuracy $\pm$ standard deviation.

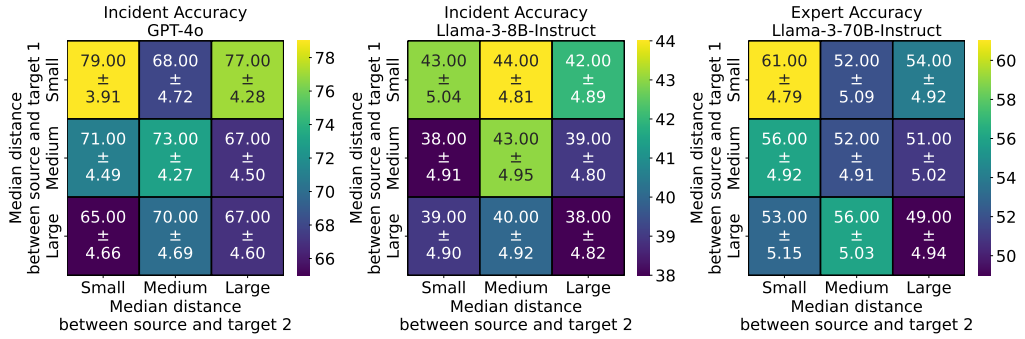

Figure 3: The effect of lost-in-distance on the similarity task is illustrated. As the distance between target node 1 and target node 2 increases, the model's accuracy degrades accordingly.

Specifically, performance at the largest distances is significantly worse compared to that at the smallest distances. As the distances increase (i.e., along the diagonal elements), the performance of all models deteriorates. In Llama-3-70B, we observe a 12% drop in model accuracy when the distance between common connections for both $|\mathcal{N}(v_i) \cap \mathcal{N}(v_j)|$ and $|\mathcal{N}(v_j) \cap \mathcal{N}(v_k)|$ increases, shifting from the $(\text{Small}, \text{Small})$ index to the $(\text{Large}, \text{Large})$ index in the heatmap plot. These results highlight that the lost-in-distance phenomenon adversely affects model performance in similarity tasks.

4 Conclusion

This study introduces the "lost-in-distance" phenomenon, demonstrating how the performance of LLMs in graph-related tasks is impacted by the relative positioning of relevant information in the input context. Our experiments reveal that as the distance between crucial data points increases, the accuracy of publicly available open and proprietary LLMs significantly deteriorates in solving graph tasks that require cross-referencing. This phenomenon compounds with the known "lost-in-the-middle" effect, leading to notable declines in task performance when multiple subgraphs must be cross-referenced, such as in common connection and similarity tasks.

Our findings, which we coin "lost-in-distance", highlight a key limitation of current LLMs in solving graph tasks, particularly those requiring complex reasoning over dispersed data. This study paves the way for future research focused on mitigating these issues, such as developing advanced graph encoding techniques. Addressing the challenges posed by lost-in-distance will be critical for enhancing LLM applications in domains like recommendation systems, molecular design, and multi-hop question answering, and will contribute to the broader field of LLMs in graph problem-solving.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. *arXiv preprint arXiv:2010.12688*, 2020.

Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. A review on language models as knowledge bases. *arXiv preprint arXiv:2204.06031*, 2022.

Nuo Chen, Yuhan Li, Jianheng Tang, and Jia Li. Graphwiz: An instruction-following language model for graph computational problems. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 353–364, 2024.

Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*, 2022.

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Paul Erdős and Alfréd Rényi. On random graphs. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Zhangyang Gao, Daize Dong, Cheng Tan, Jun Xia, Bozhen Hu, and Stan Z Li. A graph is worth k words: Euclideanizing graph using pure transformer. *arXiv preprint arXiv:2402.02464*, 2024.

Yuyao Ge, Shenghua Liu, Baolong Bi, Yiwei Wang, Lingrui Mei, Wenjie Feng, Lizhe Chen, and Xueqi Cheng. Can graph descriptive order affect solving graph problems with llms? *Authorea Preprints*, 2024.

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

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pp. 9118–9147. PMLR, 2022.

Maor Ivgi, Uri Shaham, and Jonathan Berant. Efficient long-text understanding with short-text models. *Transactions of the Association for Computational Linguistics*, 11:284–299, 2023.

Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*, 2023.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2023.

Zihan Luo, Xiran Song, Hong Huang, Jianxun Lian, Chenhao Zhang, Jinqi Jiang, and Xing Xie. Graphinstruct: Empowering large language models with graph understanding and reasoning capability. *arXiv preprint arXiv:2403.04483*, 2024.

OpenAI. Gpt-4o. `https://openai.com/index/hello-gpt-4o/`, 2023a. Accessed: 2024-04-27.

OpenAI. tiktoken: Fast bpe tokeniser for use with openai's models. `https://github.com/openai/tiktoken`, 2023b. Accessed: 2024-04-27.

Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. Let your graph do the talking: Encoding structured data for llms. *arXiv preprint arXiv:2402.05862*, 2024.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

Clayton Sanford, Bahare Fatemi, Ethan Hall, Anton Tsitsulin, Mehran Kazemi, Jonathan Halcrow, Bryan Perozzi, and Vahab Mirrokni. Understanding transformer reasoning capabilities via graph algorithms. *arXiv preprint arXiv:2405.18512*, 2024.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021.

Bin Shao, Haixun Wang, and Yatao Li. Trinity: A distributed graph engine on a memory cloud. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 505–516, 2013.

Cheng Tan, Zhangyang Gao, and Stan Z Li. Target-aware molecular graph generation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 410–427. Springer, 2023.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems*, 36, 2024.

Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, et al. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*, 2023.

Jun Xia, Chengshuai Zhao, Bozhen Hu, Zhangyang Gao, Cheng Tan, Yue Liu, Siyuan Li, and Stan Z. Li. Mole-BERT: Rethinking pre-training graph neural networks for molecules. In *The Eleventh International Conference on Learning Representations*, 2023.

Mengmei Zhang, Mingwei Sun, Peng Wang, Shen Fan, Yanhu Mo, Xiaoxiao Xu, Hong Liu, Cheng Yang, and Chuan Shi. Graphtranslator: Aligning graph model to large language model for open-ended tasks. In *Proceedings of the ACM on Web Conference 2024*, pp. 1003–1014, 2024a.

Zeyang Zhang, Xin Wang, Ziwei Zhang, Haoyang Li, Yijian Qin, and Wenwu Zhu. Llm4dyg: Can large language models solve spatial-temporal problems on dynamic graphs? In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4350–4361, 2024b.

A NOTATION

We define the distance between a common node v within two subgraphs $\mathcal{G}_u$ and $\mathcal{G}_z$ as the number of tokens separating the two occurrences of node v in the context (i.e., the textual representation of the subgraphs). The overall distance between relevant information for common connections between the two subgraphs is defined as the median of all such distances computed for each common node. Throughout the paper, we use p to indicate position and d to indicate distance.

We use accuracy, as defined below, to measure the performance of an LLM model in solving a given task:

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{\{y_i=\hat{y}_i\}} \times 100\%, \tag{1}$$

where N is the total number of samples in the task, and y_i and $\hat{y}_i$ denote the true answer and the model's answer for the ith sample, respectively. If the output of the LLM for sample i is degenerate—such as not following instructions or hallucinating—we consider it an incorrect answer, i.e., $y_i \neq \hat{y}_i$.

B LOST-IN-THE-MIDDLE FOR EDGE TASK EXISTENCE TASK

The edge existence task is analogous to the needle-in-a-haystack problem (Ivgi et al., 2023) and the document question-answering task (Liu et al., 2023), as it requires the LLM to retrieve the answer from the prompt without performing any computation. Building upon prior work in the literature by Liu et al. (2023), this study demonstrates the impact of the position of relevant information on the performance of LLMs. Specifically, it is shown that the accuracy in the edge existence task decreases when the information about the edge in question is placed in the middle of the prompt.

The prompt structure is constructed using the following procedure, which enables controlling the location of information within the prompt:

1. Randomly sample two nodes from a graph along with their corresponding connections.
2. Randomly select nine additional subgraphs and incorporate their textual encodings into the prompt. This step is necessary to examine the impact of the position of relevant information.
3. Group the subgraph structures of the two nodes of interest and position them at the beginning, middle, or end of the input context.
4. Query the model to determine whether an edge exists between the two nodes of interest.

An example of a prompt with different positions for the two nodes of interest is illustrated in Figure 4.

B.1 EXPERIMENTAL RESULTS

Lost-in-the-middle can happen in the edge existence task. To demonstrate the lost-in-the-middle phenomena in edge existence task, we experiment with one of the state of the art model as of writing this paper GPT-4 that does not use function calling. Since the lost-in-the-middle phenomenon has been extensively studied in the literature, particularly in the context of language tasks, we limit our experimentation only to GPT-4 and use its result later to study the "lost-in-distance" phenomenon.

Figure 5 shows that all encodings can cause the LLM to lose the information in the middle of the prompt. The experiment results are averaged over twenty randomly generated graph where from each graph we randomly select two nodes and form the edge existence prompt as described in previous section.

The best performance occurs when the relevant information is either at the beginning or the end of the entire subgraph structure. Even for the incident encoding which has the best performance among all encodings, the LLM still has the worst performance when the answer is located in the middle of the prompt.

You are given a graph structure in an adjacency list format.
Your task is to determine whether two given nodes are directly connected.

In this graph:
Node 208 is connected to nodes ...
Node 358 is connected to nodes ...
...
Node 425 is connected to nodes ...
Node 400 is connected to nodes ...
...
Node 714 is connected to nodes ...
Node 368 is connected to nodes ...

Question: Is node 208 directly connected to node 358?
Respond in JSON format with keys 'answer'.

(a)

You are given a graph structure in an adjacency list format.
Your task is to determine whether two given nodes are directly connected.

In this graph:
Node 425 is connected to nodes ...
Node 400 is connected to nodes ...
...
Node 208 is connected to nodes ...
Node 358 is connected to nodes ...
...
Node 714 is connected to nodes ...
Node 368 is connected to nodes ...

Question: Is node 208 directly connected to node 358?
Respond in JSON format with keys 'answer'.

(b)

You are given a graph structure in an adjacency list format.
Your task is to determine whether two given nodes are directly connected.

In this graph:
Node 425 is connected to nodes ...
Node 400 is connected to nodes ...
...
Node 714 is connected to nodes ...
Node 368 is connected to nodes ...
…
Node 208 is connected to nodes ...
Node 358 is connected to nodes ...

Question: Is node 208 directly connected to node 358?
Respond in JSON format with keys 'answer'.

(c)

Figure 4: Example of the edge existence task, illustrating the placement of the nodes of interest subgraph (nodes 208 and 358) at (a) the beginning, (b) the middle, and (c) the end of the graph structure.

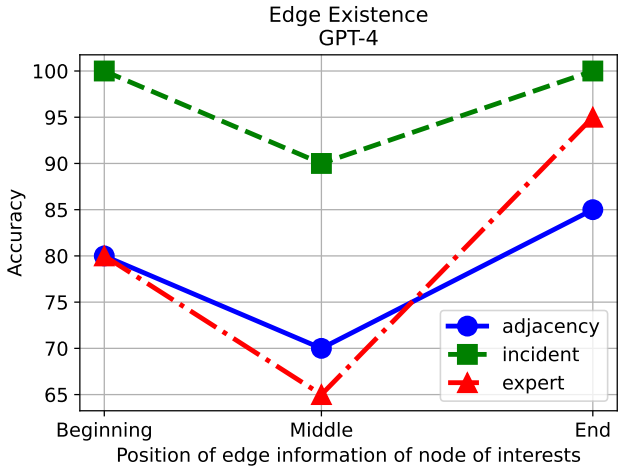

Figure 5: The effect of the position of the relevant information on the edge existence task.

C Graph Encoding

Figure 6 illustrates different techniques for the graph encoding verbalization.

D Lost-in-Distance Formulation

Tasks such as the edge existence require LLMs to perform needle-in-a-haystack retrieval, which, as previously shown, suffers from the lost-in-the-middle phenomenon in long contexts. However, in many tasks, the model not only needs to look up relevant information in the context but also requires to perform cross-referencing between retrieved information. For example, tasks like the common connection require the model to retrieve connections that *jointly* appear in both subgraphs.

We hypothesize that for tasks requiring cross-referenced retrieval, the model's performance is also impacted by the distance between relevant pieces of information, a phenomenon we term lost-in-distance. Specifically, for these tasks, the model's performance is influenced by two compounding phenomena: lost-in-the-middle when retrieving relevant information and lost-in-distance when performing a join between retrieved information.

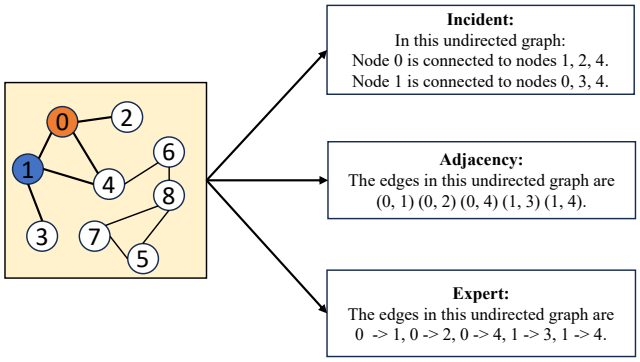

Figure 6: Three graph encoding functions, with node 0 and node 1 serving as the nodes of interest. The figure is inspired by Fatemi et al. (2024).

To explore this, we define $G(p)$ as the model's performance when the relevant information is at position p. Similarly, we define $F(p_1, p_2)$ as the model's performance when the relevant information is at positions (p_1, p_2). The value of $F(p_1, p_2)$ is estimated based on the accuracy of model in a complex task that requires cross-referencing. We hypothesize that F and G have the following relationship:

$$F(p_1, p_2) = \gamma\, G(p_1)\, G(p_2)\, H(d), \tag{2}$$

where $d = |p_2 - p_1|$ represents the distance between relevant information in the prompt and $H(d)$ represents the effect of lost-in-distance.

In the experimental section, by studying LLM performance on common connection and similarity tasks, we first demonstrate that lost-in-the-middle alone cannot explain the model's performance degradation in solving tasks that require joint reasoning across multiple subgraphs, and that it is affected by another factor, lost-in-distance.

E Analysis

E.1 Context Length

Table 1 summarizes the average context length (i.e., the number of tokens) for each task and each graph encoding. We use the tokenizer of Llama-3 to calculate the context length for Llama-3-8B-Instruct and Llama-3-70B-Instruct and use the tiktoken library (OpenAI, 2023b) to calculate the context length for GPT-4 and GPT-4o. The incident encoding produces the shortest context length, while the adjacency encoding results in the longest context length.

Table 1: Input and output context length in each encoding and each task.

Graph Task	Graph Encoding	Average Input Length	Average Output Length		
			GPT-4/4o	Llama-3-8B-Instruct	Llama-3-70B-Instruct
Edge Existence	Incident	3409.50	13	8	7
	Adjacency	6598.10	13	8	7
	Expert	5514.75	13	8	7
Common Connection	Incident	662.55	13	8	7
	Adjacency	1261.10	13	8	7
	Expert	1068.25	13	8	7
Similarity	Incident	1263.37	153.46	695.31	91.76
	Adjacency	2164.75	142.32	1126.30	106.86
	Expert	1869.29	140.34	1453.97	97.73

Table 2: Percentage of samples where each model generates degenerated answer.

Graph Encoding	Percentage of not generating the answer		
	GPT-4o	Llama-3-8B-Instruct	Llama-3-70B-Instruct
Incident	0.78%	18.78%	0.11%
Adjacency	4.11%	61.11%	0.22%
Expert	4.44%	69.00%	0.22%

E.2 Answer degeneration

LLMs sometimes fail to follow instructions and generate responses that do not adhere to the expected output template. We classify these degenerate responses as incorrect answers, i.e., $y_i \neq \hat{y}_i$ for accuracy calculation in Equation 1. Llama-3-8B-Instruct is less likely to generate a final answer compared to GPT-4o and Llama-3-70B-Instruct, which explains why it has lowest accuracy in Figure 3. Table E.2 summarizes the percentage of samples in which models fail to follow instructions. Generally, the most common patterns of degenerate answers are as follows:

- **Repetition**: LLMs sometimes repeat the same context until they reach the maximum number of output tokens, thereby failing to generate a final answer.
- **Self-contradiction**: In zero-shot step-by-step reasoning prompting (Kojima et al., 2022), LLMs are asked to answer the main question based on their responses to subquestions. However, we find that LLMs sometimes provide an incorrect final conclusion. For example, as shown in Figure 7 where we ask the LLM "*is the number of common connections between **node 658 and node 535** greater than the number of common connections between **node 535 and node 807**?*", the LLM determines that the number of common connections between node 658 and node 535 is 6, and between node 535 and node 807 is 4, but the final answer is "no" when it should be "yes".

Model Answer

<sub a>The unique common connections between node 658 and node 535 are 30, 346, 575, 605, 910, and 642.</sub a>
<sub a>6</sub a>
<sub a>The unique common connections between node 535 and node 807 are 303, 633, 642, and 92.</sub a>
<sub a>4</sub a>
<final a>No</final a>

Figure 7: Example where the answer is self-contradictory. The question in this example is that "*is the number of common connections between **node 658 and node 535** greater than the number of common connections between **node 535 and node 807**?*".

F Common Connection: More result

Figure 8 illustrates the impact of the lost-in-distance phenomenon on the GPT-4o model (OpenAI, 2023a) in solving the common connection task. Although the accuracy metrics slightly differ from those in Figure 2 for GPT-4, the pattern of the lost-in-distance effect remains consistent.

G Similarity Task

The distance distribution is illustrated in Figure 9 for three different graph encodings. We utilize the thresholds presented in Table 3 to categorize the distances into small, medium, and large groups. Furthermore, in order to make sure more uniform coverage, we employ rejection sampling to ensure that each distance group contains one hundred samples with balanced responses.

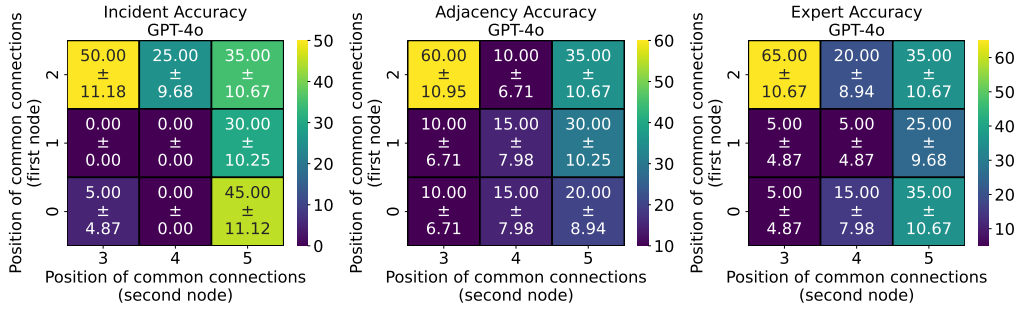

Figure 8: The effect of the position of the relevant information on the GPT-4o model solving the common connection task.

Table 3: Thresholds of each distance group for three graph encoding functions where distance is measured in number of tokens.

Graph Encoding	Small Distance	Medium Distance	Large Distance
Incident	≤ 219	$219 \sim 399$	> 399
Adjacency	≤ 425	$425 \sim 785$	> 785
Expert	≤ 354	$354 \sim 654$	> 654

To eliminate potential biases, for the three subgraphs $\mathcal{G}_{v_i}$, $\mathcal{G}_{v_j}$, and $\mathcal{G}_{v_k}$, where v_j is the source node for similarity, we generate questions randomly chosen from the following two templates:

- *Is the number of common connections between* ***node*** $\boldsymbol{v_j}$ ***and node*** $\boldsymbol{v_k}$ *greater than the number of common connections between* ***node*** $\boldsymbol{v_i}$ ***and node*** $\boldsymbol{v_j}$*?*
- *Is the number of common connections between* ***node*** $\boldsymbol{v_i}$ ***and node*** $\boldsymbol{v_j}$ *greater than the number of common connections between* ***node*** $\boldsymbol{v_j}$ ***and node*** $\boldsymbol{v_k}$*?*

G.1 Prompt Example

Using zero-shot step-by-step reasoning prompting (Kojima et al., 2022), we guide models in solving this multi-step task. Figure 10 illustrates an example of the similarity task prompt, as described in Section 3.3, along with GPT-4o's answer for solving the similarity task using incident graph encoding.

G.2 All Results

Figure 11 presents the results of the similarity task at a density of 0.1 across three models (GPT-4, Llama-8B, Llama-70B) and three different graph encodings. For all models using the graph encoding functions, we observe the typical lost-in-distance pattern, where performance at the

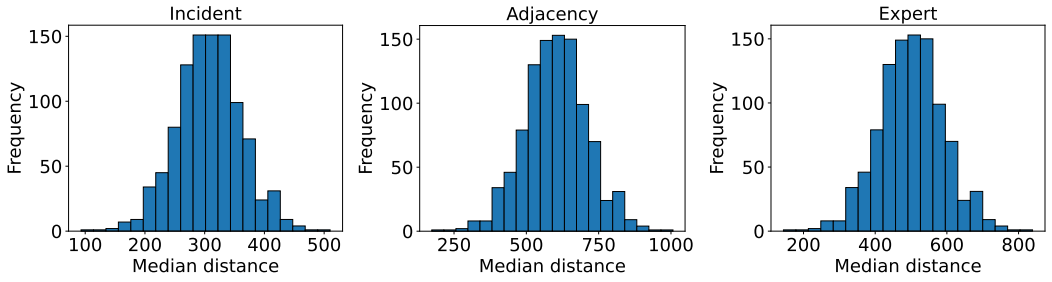

Figure 9: The distribution of median distance, in number of tokens, for three graph encoding functions.

Prompt

You are given an undirected graph structure in an adjacency list format.
Given three nodes i, j, and k, your task is to determine if the number of common connections between node i and node j is greater than the number of common connections between node j and node k.
You MUST first answer several subquestions in '<sub q></sub q>' tags in order and put your answers in '<sub a></sub a>' tags in order. Finally, answer the question in the '<final q></final q>' tag based on the given graph and your answers to the subquestions. Put your final answer in the '<final a></final a>' tag.

In this graph:
Node 963 is connected to nodes {neighbors of node 963}.
Node 452 is connected to nodes {neighbors of node 452}.
Node 423 is connected to nodes {neighbors of node 423}.

<sub q>What are the unique common connections between node 963 and node 452? Your answer MUST start with 'The unique common connections between node 963 and node 452 are'.</sub q>
<sub q>How many common connections are there between node 963 and node 452? Your answer MUST be a number only.</sub q>
<sub q>What are the unique common connections between node 452 and node 423? Your answer MUST start with 'The unique common connections between node 452 and node 423 are'.</sub q>
<sub q>How many common connections are there between node 452 and node 423? Your answer MUST be a number only.</sub q>
<final q>Based on the given graph and previous answers, is the number of common connections between node 963 and node 452 greater than the number of common connections between node 452 and node 423? Your answer MUST be 'Yes' or 'No' only.</final q>

Model Answer

<sub a>The unique common connections between node 963 and node 452 are 704, 799, 972, 430, 827, 531, 628, 496, 551, 908, 104.</sub a>
<sub a>11</sub a>
<sub a>The unique common connections between node 452 and node 423 are 104, 832, 496, 906, 824, 470, 550, 766, 634, 740.</sub a>
<sub a>10</sub a>
<final a>Yes</final a>

Figure 10: Example of the zero-shot step-by-step reasoning prompting for the similarity task.

$(\mathrm{Large}, \mathrm{Large})$ index—indicating a large distance between relevant information within two subgraphs in the prompt—is worse than at the $(\mathrm{Small}, \mathrm{Small})$ index, which indicates a smaller distance between relevant information for the similarity task in the prompt.

H Effect of Graph Density

The lost-in-distance effect remains consistent across different graph densities, i.e., different values of $P(e_{ij} \in \mathcal{E})$ in Erdős–Rényi (ER) randomly generated graphs. Graph density affects the input sequence length in a linear manner; higher densities result in proportionally longer input sequences, as demonstrated in Table 4.

Figure 12 illustrates that increasing the context length by raising graph density follows the same pattern of the lost-in-distance effect in similarity tasks. Specifically, accuracy declines progressively from top to bottom and left to right as the distances between common edges within each subgraph increase. Additionally, the figure demonstrates that for smaller context lengths, corresponding to graphs with low density, the results are noisier and the effect of lost-in-distance diminishes.

Table 4: Average Token Length from different Graph Density, $P(e_{ij} \in \mathcal{E})$, for ER graphs using expert encoding

Graph Density	**Average Token Length**
0.05	1145
0.10	1869
0.15	2633

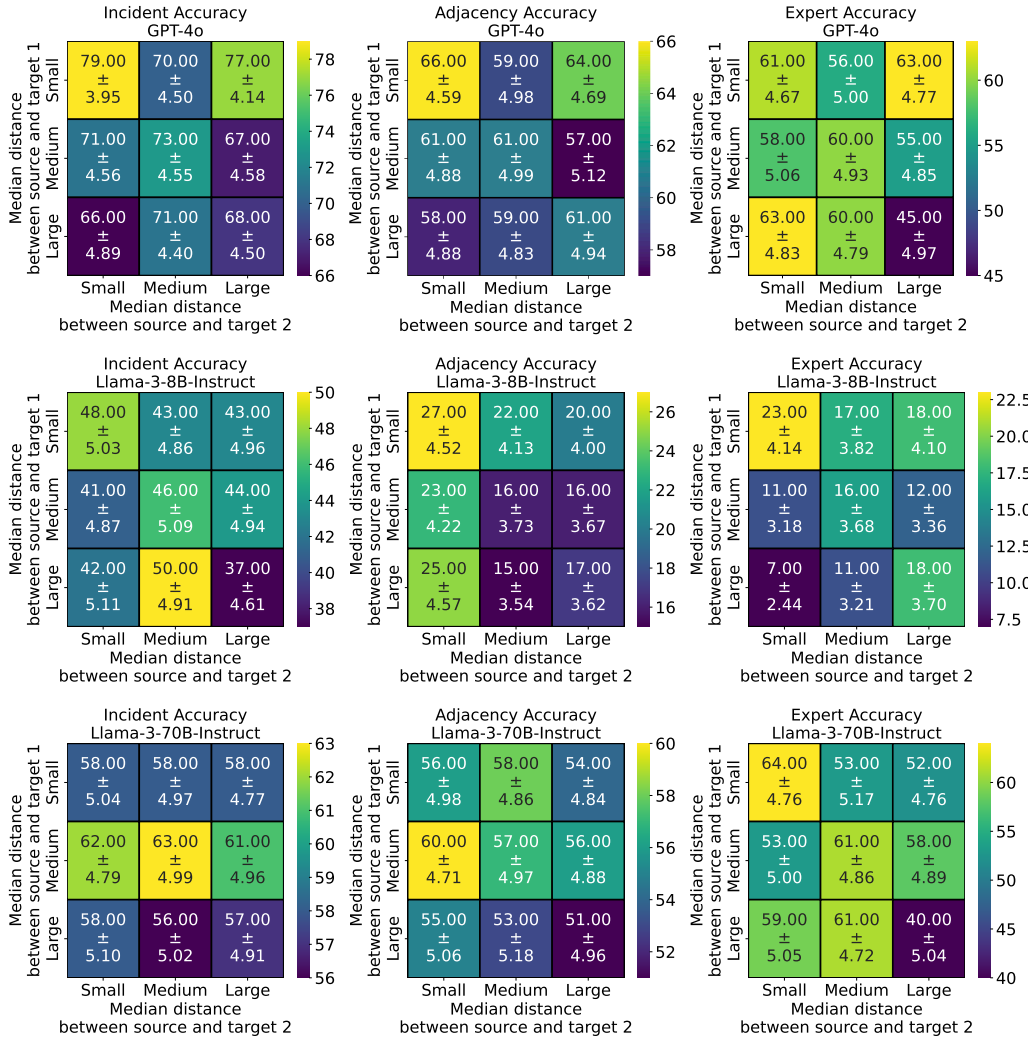

Figure 11: All results in the similarity task with density = 0.1.

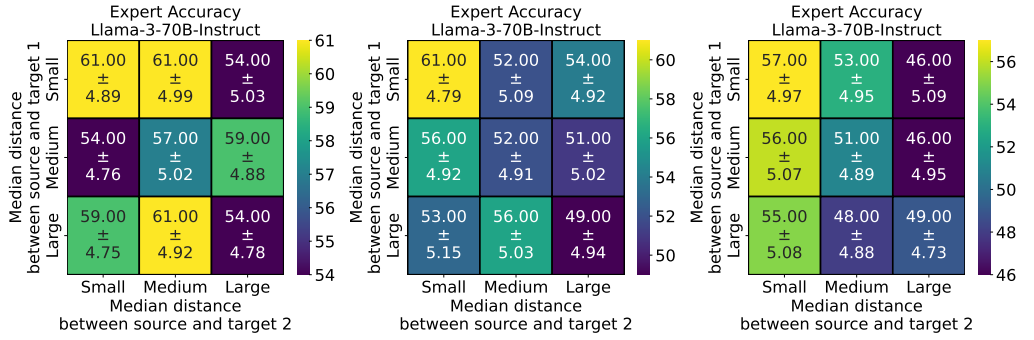

Figure 12: Results from similarity tasks with three different density values, $P(e_{ij} \in \mathcal{E})$, (left) 0.05 , (middle) 0.1, and (right) 0.15.