ReGraph: Learning to Reformulate Graph Encodings with Large Language Models

Amir Hadifar, Christopher Ochs, Arjan Van Ewijk
Nokia Bell Labs
firstname.lastname@nokia-bell-labs.com

Abstract

Large language models can rephrase and restructure natural language effectively, but their potential for reformulating graph encodings remains underexplored despite the significant impact of encoding choices on performance. In this work, we introduce ReGraph, a reinforcement learning-based approach that guides language models to reformulate graph encodings for improved task alignment. We demonstrate that reformulating graph encodings enhances reasoning and yields consistent performance gains on graph question answering tasks. Our results show that language models often prefer specific graph encodings, even if they are suboptimal for the task at hand.[1]

```
Q1: Give path from root to the node 5
Q2: How many children does node 9 have?
Q3: Number of edge in the following graph?

ASCII              | Adjacency list   | Edge list
-------------------+------------------+------------------
1                  | 0:[]             | (1→7)
|—7                | 1:[7,3,9,2]      | (1→3)
|—3                | 2:[]             | (1→9)
|  └—4             | 3:[4]            | (1→2)
|—9                | 4:[]             | (3→4)
|  |—8             | 5:[]             | (6→5)
|  |—6             | 6:[5]            | (9→8)
|  |  └—5          | 7:[]             | (9→6)
|  |—10            | 8:[]             | (9→10)
|  └—0             | 9:[8,6,10,0]     | (9→0)
└—2                | 10:[]            |
-------------------+------------------+------------------
A1: 1>9>6>5 ☑      | A1: 0>9>6>5 ☒    | A1: 1>6>5 ☒
A2: 5 ☒            | A2: 4 ☑          | A2: 5 ☒
A3: 9 ☒            | A3: 9 ☒          | A3: 10 ☑
```

Figure 1: Three questions with three different graph encodings. The language model's response varies according to the graph encoding linked to each question.

1 Introduction

Large language models (LLMs) have strong abilities to reformulate queries or prompts to improve alignment between input expressions and computational interpretation (Kong et al., 2024; Srivastava et al., 2024). However, existing reformulation techniques are largely limited to sentence-level or keyword-level transformations (Srivastava et al., 2024), falling short of leveraging the full potential of LLMs to handle more complex structural representations, such as graphs. Graphs are typically encoded in textual formats, such as edge list, adjacency list, or other notations, before being processed by LLMs (Fatemi et al., 2023). Most reformulation efforts focus on modifying the questions associated with the graphs rather than the graph encodings themselves, leaving much of their structural information untapped.

Instead, the concept of graph reformulation focuses on adapting graph encodings to facilitate more effective reasoning. Extending reformulation to the graph-level through conversions between different encodings can unlock new opportunities, as each encoding offers distinct advantages (see Figure 1), no single encoding is universally optimal (Fatemi et al., 2023; Guo et al., 2023; Jiang et al., 2025), and changing representations can serve as a powerful tool for problem solving (Korf, 1980). Yet, reformulating graphs using LLMs presents a significant challenge, as it requires generating tokens that strictly adhere to both syntactic and semantic constraints. Furthermore, this capability should be intrinsic to LLMs, as reliance on external tools or heuristics requires manual intervention. For instance, one should define that the ASCII format is more suitable for traversal problems, whereas the edge list representation is better aligned with tasks such as edge counting, thereby underscoring the need for LLMs to natively discern and adapt to appropriate graph representations.

In this study, we introduce ReGraph, a reinforcement learning (RL) method that guides LLMs to re-

[1]Code publicly available from: http:github.com/hadifar/regraph

Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 386–394

formulate graph encodings. Our approach enables models to accurately transform graph encodings, resulting in consistent performance improvements.

2 Related Work

Graph reformulation has broad application (Heckel, 2006; Hong et al., 2022), but we focus our work on LLMs that operate on graph encodings, intersecting with research in query reformulation and graph reasoning.

Query reformulation: Query and prompt rewriting methods typically refine natural language inputs through paraphrasing or iterative feedback (Lai et al., 2024; Kong et al., 2024; Liu and Mozafari, 2024; Zhu et al., 2023; Ramnath et al., 2025; Ma et al., 2023; Nogueira and Cho, 2017). However, these conversions have remained largely at the sentence level.

LLM-based graph reasoning: Traditional graph processing methods, such as graph neural networks (Fang et al., 2023; Jiang et al., 2024) and transformer-based models (Zhang et al., 2020; Kim et al., 2022), typically operate on fixed graph representations, limiting their ability to adapt to dynamic domains where structure is not fixed (Zhu et al., 2024; Wu et al., 2020). Researchers have explored LLMs as an alternative framework for graph reasoning, demonstrating strong performance (Arora et al., 2024; Fatemi et al., 2023; Wang et al., 2023). The exploration is further accelerated by the emergence of reasoning LLMs (Shao et al., 2024; Yang et al., 2025), where LLMs generate intermediate reasoning steps before producing a final answer (Jin et al., 2024; Li et al., 2025). However, most prior studies (Wang et al., 2024; Zhao et al., 2023) have concentrated on converting graphs into textual formats, rather than on transforming between different graph encodings, which is the focus of our study.

3 Methodology

3.1 Prelimanary

A *graph language model* extends conventional language modeling by conditioning token prediction on both a query and a graph represented in textual form. Given a query q and a graph encoding g corresponding to a structured graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denote the sets of nodes and edges respectively, we form the model input as $x = [q; g]$. This concatenated representation constitutes the model's prompt.

The model then generates an output y, modeling the conditional distribution $p(y \mid x)$, where y may include intermediate reasoning steps and a final answer. To evaluate the quality of a predicted output y, we define a scoring function $r(x, y) \in \mathbb{R}$ that measures its correctness or suitability with respect to the inputs. The expected accuracy of a model p is given by:

$$\text{ACCURACY}(p) = \mathbb{E}_{y \sim p(\cdot|x)} \left[r(x, y) \right]. \quad (1)$$

Given a dataset of examples, each consisting of a query, a graph, and the correct answer, our goal is to learn a model that maximizes expected performance across examples. Due to the need to optimize accuracy in discrete output spaces, recent language models (Guo et al., 2025) often employ RL methods (Sutton and Barto, 2018).

3.2 Graph Reformulation via LLM

In general, with graph reformulation, we seek to derive a graph encoding g' from the given input graph encoding g, where the new encoding preserves the properties of interest and facilitates more effective reasoning by the language model when answering the query q. In this context, g' corresponds to the reformulated encoding obtained from the model's output y. For example, given the query "`Give the path from the root to node 5`" and an input graph g encoded as an edge list, the model's output may include an alternative representation g', such as an ASCII format. This formulation adapts the initial graph encoding to better align with the requirements of the input query.

Ideally, g and g' correspond to the same underlying graph $\mathcal{G}$; however, the model may occasionally hallucinate and generate inaccurate or inconsistent reformulation. Equation 1 enables us to encourage not only the correctness of the final answer but also to promote correct graph encoding reformulation. To promote meaningful reformulation, we redefine the expectation as:

$$\mathbb{E}_{y \sim p(\cdot|x)} \left[r(x, y)\, \delta(g, g') \right], \quad (2)$$

where $\delta(g, g')$ is a measure of distance between the input graph encoding g and the inferred reformulated graph encoding g'. Specifically, we define δ as:

$$\delta(g, g') = \begin{cases} 0 & \text{if } g = g' \\ 1 - \text{Jaccard}(g, g') & \text{if } g \neq g', \end{cases} \quad (3)$$

where δ incentivizes correct reformulation by evaluating the Jaccard graph distance (Kogge, 2016) between the input and output graph encodings. It assigns a score of zero when the graph encodings are identical ($g = g'$), for example, when the LLM simply copies the input, indicating that no reformulation has occurred. The score is maximized when the distance is zero, signifying that the model has meaningfully altered the encoding while both encodings represent the same graph $\mathcal{G}$. Given that g and g' are textual representations of graphs, we parse them to obtain the nodes and edges, and compute their distance (see Appendix A for more details).

Finally, we employ Group-wise Reinforcement Policy Optimization (GRPO; Shao et al., 2024), a reinforcement learning algorithm used to finetune the LLM. GRPO encourages the LLM to generate a group of reformulations, enabling the exploration of diverse yet relevant alternatives that are optimized collectively.

4 Experiments and Analysis

4.1 Experimental Setup

With no gold standard for task selection and under computational constraints, we follow prior studies (Fatemi et al., 2023; Wang et al., 2023; Tang et al., 2025) by sampling 500 graphs across diverse graph question-answering tasks, including node count, edge count, node degree, and traversal orders (preorder, postorder, level-order), with some tasks sharing similar evaluation goals (e.g., postorder traversal corresponds to the reverse of a topological sort). Each graph is represented in three formats (edge list, adjacency list, ASCII) and three sizes: small (5–8 nodes), medium (9–16 nodes), and large (17–32 nodes), yielding a total of 27,000 instances (see Appendix B for details).

The base language models used for the evaluation include Qwen3-8B, Qwen3-32B (Yang et al., 2025), Gemma3-12B (Team et al., 2025), and Ministral-8B (Mistral AI team, 2024). We used 10% of the dataset for reinforcement learning finetuning (RLF) and the remaining 90% for evaluation. Specifically, we applied RLF to Qwen3-8B variants: ReGraph (Qwen3-8B-RLF) with both reformulation and accuracy objectives, and Qwen3-8B-RLF with the accuracy objective.

For evaluation, we investigate three aspects: i) **Graph reformulation** evaluates the models' ability to perform reformulation across various types

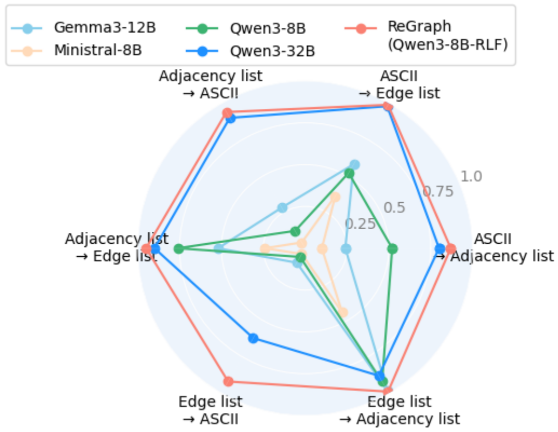

Figure 2: Reformulation capabilities of different LLMs.

of reformulation. In this setting, we provide the LLMs with graphs represented in a specific encoding and instruct them to reformulate these representations into a designated target encoding. ii) **Graph question answering** assesses the effectiveness of graph reformulation in solving graph question answering tasks defined above. We compare model performance before and after finetuning to quantify the benefits introduced by reformulation. iii) **Encoding preferences** evaluate the models' choices when instructed to select an encoding from a predefined set. LLMs are given graph data in `GraphML` format (Brandes et al., 2001) and asked to convert it into one of several alternative encodings: ASCII, adjacency list, or edge list. Details on prompt formatting for each setting are provided in Appendix C. Accuracy is used as an evaluation metric for all experiments.

4.2 Result and Disucssion

Graph reformulation: Figure 2 shows the performance of various models in reformulating graph encodings. ReGraph demonstrates consistently strong results, successfully transforming representations in most cases and even outperforming larger models like Qwen3-32B. The results also reveal a trend where reformulation capability improves with model size, as seen in the performance gap between Qwen3-8B and Qwen3-32B. ASCII-based representations remain challenging for most models, likely due to limited exposure during pretraining or finetuning. Conversely, nearly all models achieve near-perfect accuracy when converting between edge list and adjacency list (e.g., Edge list $\rightarrow$ Adjacency list).

Graph question answering: Figure 3 illustrates

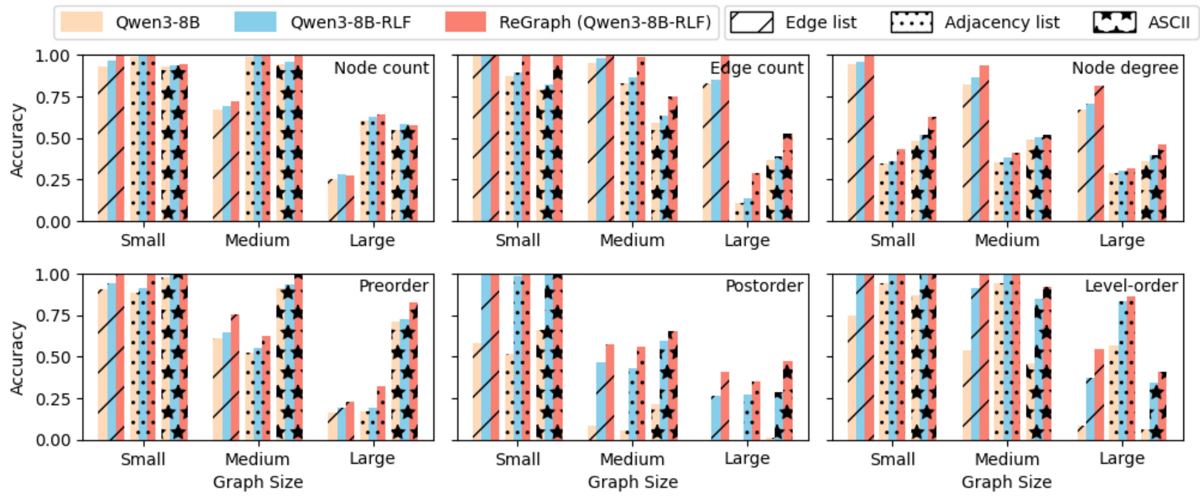

Figure 3: Performance comparison of different models across varying graph problem and graph sizes.

our results on graph question answering. Different graph encodings offer distinct advantages depending on the task. For instance, edge list encodings are particularly effective for edge counting, whereas adjacency list encodings yield better performance on node degree tasks. These results underscore the importance of selecting task-appropriate graph representations and highlight the potential of graph reformulation to enhance model performance. ReGraph improves accuracy across nearly all tasks, regardless of the input encoding, when compared to baseline models. However, the benefit of reformulation varies by task. For edge-encoded inputs, traversal tasks (e.g., postorder, level-order) show consistent gains, while improvements on node degree tasks are minimal. Notably, when edge list encodings are used for traversal tasks, performance often improves as the model frequently reformulates the input into more structured formats such as ASCII or adjacency list. Qwen3-8B-RFT exhibits performance comparable to ReGraph, albeit slightly lower, with both results obtained via direct RLF, and additional data or initial supervised fine-tuning likely to further improve performance (Yeo et al., 2025).

Encoding Preferences: As illustrated in Figure 4, most LLMs exhibit a strong preference for either edge list or adjacency list formats. For instance, Llama3.1-70B and Gemma3-12B predominantly favor adjacency list, whereas Qwen models typically prefer edge list. This tendency likely reflects the graph encoding most commonly encountered during pretraining or finetuning. However, these favored encodings are not always the most effective for every task. For example, our earlier experiments demonstrated that ASCII encodings yield superior performance for preorder traversal, yet they remain underutilized by LLMs. Addressing these biases presents an opportunity to enhance downstream task performance.

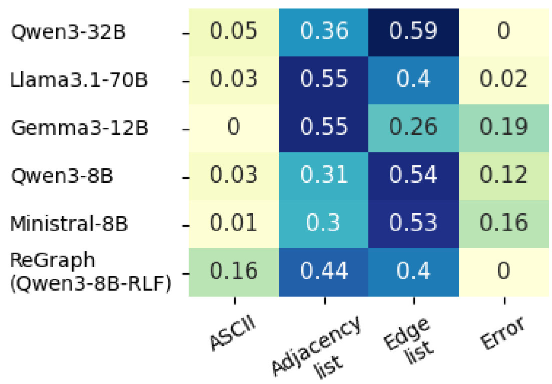

Figure 4: Comparison of biases exhibited by different LLMs when tasked with converting graph data in `GraphML` (Brandes et al., 2001) into one of the alternative graph encodings.

5 Conclusion

We propose ReGraph, an RL-based method that guides language models to reformulate graph encodings into more effective representations. Our model is finetuned using reinforcement learning to jointly optimize question answering and encoding reformulation. Experiments on graph question answering demonstrate consistent improvements in both reasoning and reformulation quality.

Limitations

Although ReGraph delivers promising results, several important limitations remain. Our current work focuses on labeled graphs, which limits its applicability to other types such as unlabeled, weighted, or dynamic graphs. Extending reformulation techniques to these graph classes is a valuable direction for future research. Additionally, our experiments are conducted solely with the Qwen3 model and three graph encodings due to resource constraints. While we expect the findings to generalize, broader evaluation across diverse LLMs and graph encodings is necessary to fully validate this assumption.

References

Zeina Abu-Aisheh, Romain Raveaux, Jean-Yves Ramel, and Patrick Martineau. 2015. An exact graph edit distance algorithm for solving pattern recognition problems. In *4th International Conference on Pattern Recognition Applications and Methods 2015*.

Jashn Arora, Rahul Madhavan, Karthikeyan Shanmugam, John Palowitch, and Manish Jain. 2024. Treetop: Topology-aware fine-tuning for llm conversation tree understanding. In *NeurIPS 2024 Workshop on Fine-Tuning in Modern Machine Learning: Principles and Scalability*.

Ulrik Brandes, Markus Eiglsperger, Ivan Herman, Michael Himsolt, and M Scott Marshall. 2001. Graphml progress report structural layer proposal: Structural layer proposal. In *International Symposium on Graph Drawing*, pages 501–512. Springer.

Taoran Fang, Yunchao Zhang, Yang Yang, Chunping Wang, and Lei Chen. 2023. Universal prompt tuning for graph neural networks. *Advances in Neural Information Processing Systems*, 36:52464–52489.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2023. Talk like a graph: Encoding graphs for large language models. *arXiv preprint arXiv:2310.04560*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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

Aric Hagberg, Pieter J Swart, and Daniel A Schult. 2008. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States).

Reiko Heckel. 2006. Graph transformation in a nutshell. *Electronic notes in theoretical computer science*, 148(1):187–198.

Jiwon Hong, Dong-hyuk Seo, Jeewon Ahn, and Sang-Wook Kim. 2022. Graphreformcd: Graph reformulation for effective community detection in real-world graphs. In *Companion Proceedings of the Web Conference 2022*, pages 180–183.

Xinke Jiang, Rihong Qiu, Yongxin Xu, Yichen Zhu, Ruizhe Zhang, Yuchen Fang, Chu Xu, Junfeng Zhao, and Yasha Wang. 2024. Ragraph: A general retrieval-augmented graph learning framework. *Advances in Neural Information Processing Systems*, 37:29948–29985.

Zhuohang Jiang, Pangjing Wu, Ziran Liang, Peter Q Chen, Xu Yuan, Ye Jia, Jiancheng Tu, Chen Li, Peter HF Ng, and Qing Li. 2025. Hibench: Benchmarking llms capability on hierarchical structure reasoning. *arXiv preprint arXiv:2503.00912*.

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and 1 others. 2024. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. *arXiv preprint arXiv:2404.07103*.

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

Peter M Kogge. 2016. Jaccard coefficients as a potential graph benchmark. In *2016 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*, pages 921–928. IEEE.

Weize Kong, Spurthi Amba Hombaiah, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024. Prewrite: Prompt rewriting with reinforcement learning. *arXiv preprint arXiv:2401.08189*.

Richard E Korf. 1980. Toward a model of representation changes. *Artificial intelligence*, 14(1):41–78.

Yilong Lai, Jialong Wu, Congzhi Zhang, Haowen Sun, and Deyu Zhou. 2024. Adacqr: Enhancing query reformulation for conversational search via sparse and dense retrieval alignment. *arXiv preprint arXiv:2407.01965*.

Qianlong Li, Chen Huang, Shuai Li, Yuanxin Xiang, Deng Xiong, and Wenqiang Lei. 2025. GraphOTTER: Evolving LLM-based graph reasoning for complex table question answering. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5486–5506, Abu Dhabi, UAE. Association for Computational Linguistics.

Jie Liu and Barzan Mozafari. 2024. Query rewriting via large language models. *arXiv preprint arXiv:2403.09060*.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315.

Mistral AI team. 2024. Un mistral, des ministraux. https://mistral.ai/news/ministraux/. Accessed: 2025-07-28.

Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. *arXiv preprint arXiv:1704.04572*.

Kiran Ramnath, Kang Zhou, Sheng Guan, Soumya Smruti Mishra, Xuan Qi, Zhengyuan Shen, Shuai Wang, Sangmin Woo, Sullam Jeoung, Yawei Wang, and 1 others. 2025. A systematic survey of automatic prompt optimization techniques. *arXiv preprint arXiv:2502.16923*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Saurabh Srivastava, Chengyue Huang, Weiguo Fan, and Ziyu Yao. 2024. Instances need more care: Rewriting prompts for instances with LLMs in the loop yields better zero-shot performance. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6211–6232, Bangkok, Thailand. Association for Computational Linguistics.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press Cambridge.

Jianheng Tang, Qifan Zhang, Yuhan Li, Nuo Chen, and Jia Li. 2025. Grapharena: Evaluating and exploring large language models on graph computation. In *The Thirteenth International Conference on Learning Representations*.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems*, 36:30840–30861.

Zehong Wang, Sidney Liu, Zheyuan Zhang, Tianyi Ma, Chuxu Zhang, and Yanfang Ye. 2024. Can llms convert graphs to text-attributed graphs? *arXiv preprint arXiv:2412.10136*.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*.

Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. 2020. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*.

Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. 2023. Graphtext: Graph reasoning in text space. *arXiv preprint arXiv:2310.01089*.

Kerui Zhu, Bo-Wei Huang, Bowen Jin, Yizhu Jiao, Ming Zhong, Kevin Chang, Shou-De Lin, and Jiawei Han. 2024. Investigating instruction tuning large language models on graphs. *arXiv preprint arXiv:2408.05457*.

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zheng Liu, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107*.

A Graph distance

We initially used graph edit distance (Abu-Aisheh et al., 2015) to measure similarity between different graph encodings. However, due to the high computational cost of existing implementations, taking over 300 seconds for certain graph pairs, this approach quickly became a bottleneck in our pipeline. To address this, we switched to the Jaccard distance, which offers significantly better computational efficiency (Kogge, 2016). Our implementation of the Jaccard distance is as follows:

```
Jaccard(G1, G2):
  N1 <- NodeSet(G1)
  N2 <- NodeSet(G2)
  E1 <- EdgeSet(G1)
  E2 <- EdgeSet(G2)
  J_edges <- |E1 intersect E2| / |E1 union E2|
  J_nodes <- |N1 intersect N2| / |N1 union N2|
  return (J_nodes + J_edges) / 2
```

B Dataset and Training Details

We adopt the methodology of (Fatemi et al., 2023; Wang et al., 2023; Tang et al., 2025) to generate random graphs using the NetworkX library (Hagberg et al., 2008) and to compute the ground truth for various graph-based tasks. While there is no gold standard for task selection, we focused on well-known tasks with rich real-world applications that assess multi-dimensional reasoning skills. The tasks considered include: (1) *Node count*: determining the total number of nodes in the graph, (2) *Edge count*: determining the total number of edges in the graph, (3) *Node degree*: computing the degree of a specified node, (4) *Preorder*: performing a traversal in root-left-right order, (5) *Postorder*: performing a traversal in left-right-root order, (6) *Level-order*: breadth-first traversal that visits nodes sequentially by levels starting from the root.

For training models, we used four NVIDIA H100 GPUs to run our experiments. Our model was finetuned using the official hyperparameter recommendations provided by Qwen[2]. We used the `AdamW-8bit` optimizer with a learning rate of 1e-6. Training was conducted for one epoch with a batch size of 2 and a gradient accumulation factor of 2. The maximum input length was set to 1024 tokens, and the maximum output generation length was 6144 tokens. All experiments were conducted using Hugging Face TRL[3], and training took approximately 18 hours in total.

C Prompts

This appendix provides representative prompt examples used in each of our evaluation settings: i) graph reformulation (Figure 5), ii) downstream task performance (Figure 6), and iii) encoding preference (Figure 7).

[2]https://qwen.readthedocs.io/en/latest
[3]https://huggingface.co/docs/trl/en/index

Prompt

```
User asks a graph related question. The assistant first thinks about the reasoning process in the
mind and then provides the user with the answer. The assistant can convert graph structures to
better answer the final question, converting between formats: {list_of_encodings}
The reasoning process are enclosed within <think> and </think> tags.
The transformation must enclose within <graph> and </graph> tags.
The final answer is enclosed within <answer> and </answer> tags.

Q: Given a labeled directed graph in ASCII format, What is the degree of node 5?
1
├── 2
│   └── 4
│       └── 5
└── 3
```

Example output

```
<think> To compute the degree of node 5, it's helpful to convert the ASCII format into an edge
list.
<graph>
(1,2)
(1,3)
(2,4)
(4,5)
</graph>
Now, using the edge list, I will calculate the degree of node 5. Since the graph is directed, we
compute: Total degree = in-degree + out-degree = 1 + 0 = 1 </think>

<answer>1</answer>
```

Figure 5: An example prompt and its corresponding output for the node degree task.

Prompt

```
You are given a graph in a specific encoding format. Your task is to convert this graph into
another encoding format as specified by the user.
The transformed graph must enclose in <graph> </graph> tags.
Return the transformed graph without any explanation.

Here are possible encodings format:

Encoding-1 (ASCII):      Encoding-2 (edge list):     Encoding-3 (adjacency list)
a                        (a,b)                       a: [b]
└── b                    (b,c)                       b: [c,d]
    ├── c                (b,d)                       c: []
    └── d

User: Convert the following graph from ASCII encoding to edge list encoding.
a
├── b
├── c
└── d
```

Example output

```
<graph>
  (a,b)
  (a,c)
  (a,d)
</graph>
```

Figure 6: Sample prompt and corresponding output for graph reformulation.

Prompt

```
Convert the user's graph into one of the following encodings:

Encoding-1 (ascii):
a
└── b
    ├── c
    └── d

Encoding-2 (edge list):
(a,b)
(b,c)
(b,d)

Encoding-3 (adjacency list)
a: [b]
b: [c,d]
c: []

Return the converted graph without any explanation.

User graph:
<graph id="G" edgedefault="undirected">
    <node id="a"/>
    <node id="b"/>
    <node id="c"/>
    <node id="d"/>
    <node id="f"/>
    <edge id="e0" source=c target=d/>
    <edge id="e1" source=d target=f/>
    <edge id="e2" source=f target=a/>
    <edge id="e3" source=a target=b/>
</graph>
```

Example output

```
(c,d)
(d,f)
(f,a)
(a,b)
```

Figure 7: Example prompt with GraphML input and a sampled edge list output, illustrating the model's tendency to favor specific formats such as edge list when reconstructing graph structures.