RoGTR: Robust Table Question Answering via Graph-based Textual Reasoning

Xiaoke Guo
School of Software Technology
Zhejiang University
Hangzhou, China

Wen Zhang*
Zhejiang University
ZJU-Ant Group Joint Lab of Knowledge Graph
Hangzhou, China

***Abstract*—Large language models (LLMs) have advanced table-based question answering (Table QA), but current approaches face a critical trade-off. Symbolic reasoning methods, such as addressing specific questions via SQL generation, offer precision but are brittle, often failing on structurally complex or irregular tables. Conversely, purely End-to-End textual reasoning handles flexibility but struggles with context overload, poor structural comprehension, and computational inaccuracies—key limitations identified in complex QA scenarios. To bridge this gap, we introduce Structured Textual Reasoning, a new paradigm that integrates the precision of symbolic methods with the flexibility of textual inference. We present RoGTR (Robust Graph-based Textual Reasoning), a framework designed to overcome these limitations. RoGTR begins by converting diverse tables into a unified, triple-based textual representation. This structured format is then processed by LLM using a dual-channel reasoning strategy. This representation allows the model to effectively filter irrelevant information and infer latent semantic relationships, thereby addressing the core challenges of Table QA without resorting to fragile, hard-coded parsers or code generation. Extensive experiments demonstrate that RoGTR achieves state-of-the-art performance on complex table benchmarks while maintaining robust accuracy on simple tables.**

***Index Terms*—Table Question Answering; Large Language Models; Table Reasoning**

I. Introduction

Tables serve as a dense repository of structured knowledge across the Web and databases. Consequently, they are increasingly studied for table question answering (TableQA). Tabular data is structurally diverse, categorized into flat and complex tables [1, 2]. Flat tables feature fixed columns with rows as single entities; complex tables include hierarchical structures and merged cells, offering flexibility but posing parsing challenges. The remarkable progress of large language models (LLMs) in diverse Natural Language Processing tasks has spurred significant research interest in applying their potent capabilities to tabular question answering [3, 4].

While LLMs have demonstrated significant potential for Table QA [5], their application is hindered by several key challenges: (1) **Context Overload**, where voluminous or noisy inputs lead to model distraction, hallucinations, and diminished instruction following ability; (2) **Lack of Structural Awareness**, as LLMs struggle to infer latent semantic relationships beyond the table's explicit layout, limiting deep comprehension; and (3) **Computational Limits**, with models often failing to correctly execute the multi-step arithmetic and logical operations required for complex reasoning.

Existing solutions offer partial remedies but entail significant trade-offs. To mitigate Context Overload (1), symbolic reasoning methods [2, 6, 7] generate external code (e.g., SQL, Python) to filter for relevant information. However, these methods are typically constrained to relational tables with fixed schemas, limiting their effectiveness on the complex, heterogeneous structures often found in real-world data. To enhance Structural Awareness (2), certain approaches incorporate specialized table encoders [8, 9] or fine-tune models on tabular data [10]. These methods, however, necessitate extensive training and lack adaptivity to emerging state-of-the-art LLMs. To address Computational Limits (3), code generation is employed to facilitate multi-step reasoning. Yet, this approach relies heavily on data normalization; for example, symbolic execution on unnormalized fields can lead to errors, such as lexicographically sorting string-formatted currency (e.g., ranking "900"$above$"100,000"). Furthermore, To integrate the exactness of symbolic execution with the resilience of textual reasoning against unnormalized data structures, hybrid reasoning [11, 12] has been proposed as a compelling solution.

Based on these limitations, we argue that the promising path is not to depend on brittle symbolic parsers, but to instead harness the textual reasoning capabilities of LLMs. Accordingly, we propose **RoGTR**, a **Ro**bust **G**raph-based **T**extual **R**easoning approach designed to achieve state-of-the-art (SOTA) performance across diverse TableQA tasks. RoGTR combines lightweight preprocessing with calculator-augmented textual reasoning, effectively overcoming the limitations of TableQA.In summary, this paper contributes:

- We introduce RoGTR, a novel framework designed to enhance textual reasoning in tabular QA. RoGTR utilizes a graph-based serialization approach to effectively manage the structural diversity and complexity of real-world tables.
- We propose a new reasoning paradigm for tableQA that addresses key challenges by mining latent semantic information and incorporating a dual-channel reasoning strategy.
- We conduct extensive experiments showing that RoGTR achieves SOTA performance on both complex table benchmarks and simpler tables.

*Corresponding author.

II. Related Work

A. Table Serialization

Due to their sequence-to-sequence architecture, LLMs require tabular data to be serialized into linear text [3, 13]. Common serialization strategies employ markup formats, such as HTML, Markdown and natural language delimiters. Embedding-based alternatives utilize table-specific encoders [14, 15] or specialized LLM variants [8]. Distinct from text-based or embedding-based methods, representing tables as graphs or trees has emerged as a promising approach. For instance, recent works [5] propose transforming tables into tree structures. Similarly, GraphOTTER [16] maps complex tables onto undirected graphs. Conversely, we leverage directed graphs to enable fine-grained representation.

B. Table Reasoning

Table reasoning has seen methodologies evolve across pre-LLM and LLM eras. Traditional approaches primarily utilized supervised fine-tuning [8, 17]. The advent of LLMs introduced capabilities like in-context learning [18, 19]. Instruction design further enhanced the generalization capability of LLM to unseen tasks by decomposing table reasoning or invoking external tools [20, 21]. Moreover, step-by-step reasoning has gained prominence, allowing models to solve complex queries through intermediate steps [22–24]. These advancements signify a transition towards more flexible, generalizable, and interpretable table reasoning.

III. Method

A. Task Definition

Let $\mathbf{Q} = (q_1, \ldots, q_n)$ be a sequence of tokens representing a query, and let $\mathbf{P}$ be an optional context prompt. The task involves reasoning over two primary table structures:

1) Flat Table: A flat table is denoted by the tuple:

$$\mathbf{T}_{\text{flat}} = (\mathbf{V}, \mathbf{\Sigma}_c) \tag{1}$$

where $\mathbf{V} \in \mathbb{R}^{R \times C}$ is the data matrix, with R rows and C columns. $\mathbf{\Sigma}_c = (c_1, \ldots, c_C)$ is the set of column headers.

2) Complex Table: A complex, or hierarchical, table is represented as:

$$\mathbf{T}_{\text{hier}} = (\mathbf{V}, H_{\text{top,left}}, \mathcal{R}_{\text{merge}}) \tag{2}$$

where $\mathbf{V} \in \mathbb{R}^{M \times N}$ is the data matrix. $H_{\text{top,left}}$ denotes the hierarchical headers. $\mathcal{R}_{\text{merge}}$ is the set of merged cell regions.

3) Task Objective: Given a table $\mathbf{T} \in \{\mathbf{T}_{\text{flat}}, \mathbf{T}_{\text{hier}}\}$ and a query $\mathbf{Q}$, the objective is to generate an answer sequence $\mathbf{A}$:

$$\mathbf{A} = M_\theta(\mathbf{T}, \mathbf{Q}, \mathbf{P}) \tag{3}$$

B. Overall Framework

We introduce **RoGTR**, a framework for question answering over diverse table structures. It integrates (1) graph-based table serialization, (2) optimized information retrieval, (3) latent relationship mining, and (4) a joint reasoning strategy.

1) Graph Construction and Refinement:

1. Table Serialization: RoGTR first serializes an input table $\mathbf{T}$ into a directed graph $G = (N, E)$, where the nodes' row and column indices inherently encode the table's grid structure. This encoding implicitly enables the LLM to comprehend the precise position and structural context of each cell. The edges $E_{ij} \in E$ are represented as directed tuples (N_i, N_j, R_{ij}), where R_{ij} denotes an implicit spatial relationship (e.g., `is_right_of`, `is_below`) inferred solely from the relative cell positions. In contrast to conventional methods that require pre-annotated headers or fixed header locations [11, 22, 25], our graph construction does not distinguish between header and data cells. This unified approach simplifies graph generation, enhances robustness against diverse table layouts, and obviates the need for explicit cell-type annotation.

2. Tuple Selection: Following serialization, a filtering stage identifies a subgraph of key cells relevant to the query to mitigate context overload. This process begins with *query optimization*. The raw user query Q often contains implicit references or vague categorical terms (e.g., asking about "items in this category" without specifying them) that hinder effective dense retrieval. To bridge this gap, an LLM—guided by a prompt P_{opt}—refines Q into a semantically enriched version, Q_{opt}. As shown in Equation 4, Q_{opt} concretizes vague terms into explicit schema elements or candidate entities found in the table $\mathbf{T}$ (Figure 2):

$$Q_{opt} = \text{LLM}(Q, \mathbf{T}, P_{opt}) \tag{4}$$

This step ensures that the retriever can accurately match the query against specific table contents. Crucially, to mitigate the risk of this optimization introducing bias or altering the query's original intent, Q_{opt} is used *exclusively* for this filtering step. The subsequent final reasoning stage reverts to the original, unmodified query Q to ensure fidelity. Next, a dense retrieval function F uses Q_{opt} to compute the semantic similarity between the query and the textual value of each node in the full graph G. These nodes form the initial key subgraph, G_{key}:

$$G_{key} = F(G, Q_{opt}) \tag{5}$$

Finally, to ensure contextual completeness, this initial subgraph is expanded by including all nodes from G that share a row or column with any node in G_{key}. This process yields the final filtered subgraph for downstream reasoning.

3. Semantic Enhancement: This semantic enhancement stage is designed to address the key challenge of poor structural comprehension, where LLMs often fail to infer relationships beyond a table's explicit layout. The initial subgraph G_{key} captures only spatial relationships (e.g., adjacency), failing to make explicit the complex hierarchical structures that these spatial relations imply. For instance, a cell 'Total Number' may be spatially `is_above` 'Female' and 'Male', but G_{key} only registers this adjacency, missing the latent semantic hierarchy (i.e., that 'Female' is a sub-category of 'Total Number'). Furthermore, complex tables often contain latent semantic relationships (e.g., hierarchies, logical connections) that are crucial for reasoning but may span non-adjacent cells. To uncover this deeper structure, we use LLM

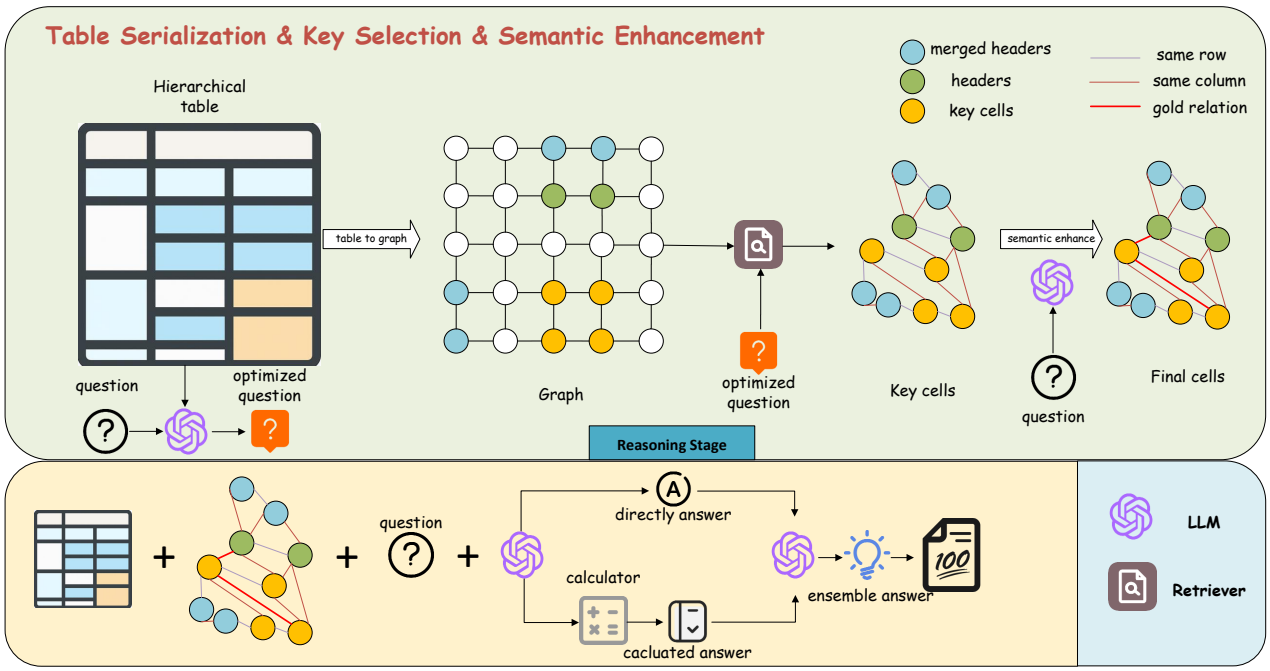
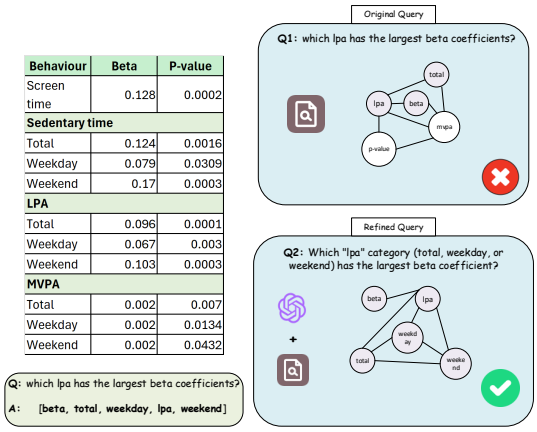


Fig. 1. Overview of RoGTR.



Behaviour	Beta	P-value
Screen time	0.128	0.0002
Sedentary time		
Total	0.124	0.0016
Weekday	0.079	0.0309
Weekend	0.17	0.0003
LPA		
Total	0.096	0.0001
Weekday	0.067	0.003
Weekend	0.103	0.0003
MVPA		
Total	0.002	0.007
Weekday	0.002	0.0134
Weekend	0.002	0.0432

Fig. 2. LLM-Empowered Query Optimization: Enhancing Cell Localization Accuracy in Dense Retrieval by Understanding Global Table Structure

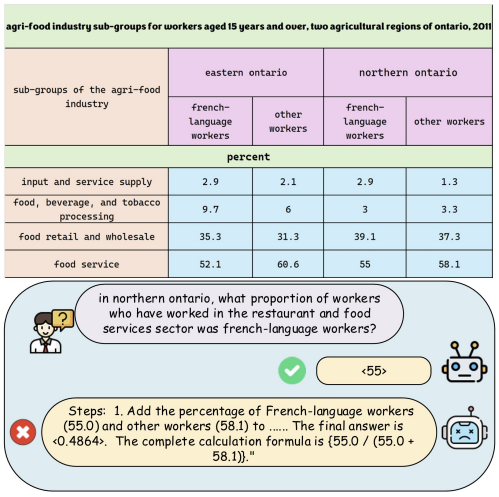


Fig. 3. The Advantage of Mixed Thinking: When "Fast Thinking" is Superior to "Slow Thinking"

to mine these latent semantic relationships and generate a set of supplementary, semantically-enhanced edges, E_{enrich}:

$$E_{enrich} = \text{LLM}(G_{key}, P_{enrich}, Q) \tag{6}$$

Here, the prompt P_{enrich} guides the LLM to analyze the nodes and structure of G_{key} in the context of the query Q. The final graph for reasoning, G_{final}, is formed by augmenting the subgraph with these new edges:

$$G_{final} = (N_{key}, E_{key} \cup E_{enrich}) \tag{7}$$

where N_{key} and E_{key} are the nodes and edges from G_{key}.

2) Reasoning Stage: As observed in case studies (Figure 3), applying complex, few-shot reasoning to simple lookup queries can lead to unnecessary complexity and cumulative errors. To address this, we employ a mixed reasoning strategy with two distinct thinking modes, diverging from a primary focus on symbolic versus textual distinctions:

1) **Direct Mode:** The LLM performs a direct lookup on the graph to generate an immediate answer, A_1. This mode is fast and effective for simple queries.
2) **CoT Mode:** The LLM engages in step-by-step reasoning, leveraging in-context examples and an external calculator to derive a more considered answer, A_2. This mode is designed for complex queries requiring calculations or multi-step logic.

The final answer is determined via a self-correction mechanism. If the answers from both modes are identical ($A_1 = A_2$), this consistent result is returned. If they diverge, the LLM initiates a self-reflection step, using a dedicated prompt P_{sel}

to arbitrate between the conflicting answers and produce the final response. This logic is formalized as:

$$A_{final} = \begin{cases} A_1 & \text{if } A_1 = A_2, \\ \text{LLM}(A_1, A_2, T, Q, P_{\text{sel}}) & \text{if } A_1 \neq A_2 \end{cases} \quad (8)$$

For a Self-Consistency ensemble, multiple inference runs are performed, and a majority vote over the outcomes determines the final answer.

IV. Experiment

A. Experimental Setup

Dataset: We evaluate on HiTab [25] and AIT-QA [26] for complex tables, and WikiTQ [27] for flat tables.

Backbones: For the Tuple Selection stage (information retrieval), `gte-Qwen2-7B`[1] served as the primary text embedding model for generating node and query vectors. For the subsequent Reasoning Stage, we selected 4 LLMs as backbones, encompassing both open- and closed-source options with varying parameter scales. For the closed-source models, we employed `GPT-4o-mini` [28] as a representative smaller model and `GPT-4o` [29] as a state-of-the-art large model. As corresponding open-source counterparts, we utilized `Qwen2.5-32B` [30] and `Phi-4-14B` [31].

Baseline: We selected 4 competitive LLM-based approaches as baselines: **(1) Zero-shot** [32]: Providing the LLM with the table, query, and basic task instructions in a single prompt. **(2) EEDP** [23]: Utilizes step-by-step prompts to guide LLM inference for acquiring domain knowledge, identifying key information, and decomposing complex calculations into atomic operations.**(3) E5** [33]: Employs a structured five-stage procedure: Explain, Extract, Execute, Exhibit, and Extrapolate.**(4) GraphOTTER** [16]: Serializes the input table into triples and uses an iterative Thought-Action cycle to update the reasoning state and trace, leveraging Chain-of-Thought (CoT) prompts for final answer generation based on the preceding reasoning.

Metrics: Following established practices in prior work [5], we employ **Exact Match (EM) accuracy** as the primary evaluation metric. This metric assesses the performance by verifying if the predicted answer exactly corresponds, character by character, to the ground truth label. To ensure consistent and programmatically verifiable outputs suitable for this strict evaluation, the LLMs were specifically instructed via prompts to generate their answers adhering to a predefined format.

B. Main Results

Table I presents the results on Hitab and AIT-QA. RoGTR consistently achieves SOTA performance.

Superiority of Structured Representation: RoGTR and GraphOTTER significantly outperform E5, suggesting structured formats like graphs provide a more digestible input for LLMs than raw text.

Robustness to Model Scale: While GraphOTTER approaches RoGTR's performance on large models, the gap widens on smaller models, highlighting RoGTR's efficiency.

[1]Available at https://huggingface.co/Alibaba-NLP/gte-Qwen2-7B-instruct

TABLE I
PERFORMANCE COMPARISON ON HITAB AND AIT-QA (EXACT MATCH). BEST IN **BOLD**, SECOND IN UNDERLINED.

Method	HiTab — Exact Match (EM) ↑				AIT-QA — Exact Match (EM) ↑			
	4o-mini	4o	Qwen	Phi	4o-mini	4o	Qwen	Phi
Zero-shot	0.603	0.747	0.775	0.677	0.664	0.784	0.805	0.770
EEDP	0.650	0.797	0.763	0.710	0.803	0.856	0.775	0.744
E5	0.265	0.793	–	–	0.235	0.871	–	–
GraphOTTER	0.688	0.834	0.813	0.728	0.813	0.881	0.827	0.767
RoGTR	**0.745**	**0.858**	**0.838**	**0.759**	**0.855**	**0.897**	**0.838**	**0.794**

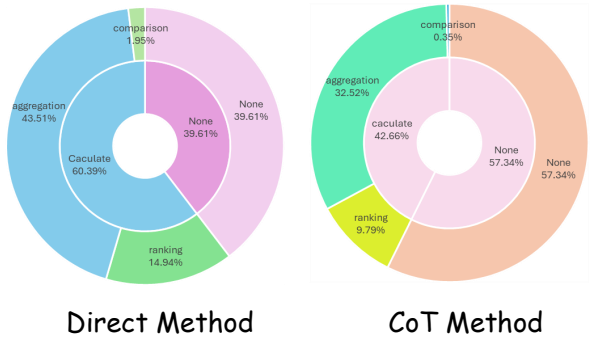


Fig. 4. Error Distribution: Direct vs. Chain-of-Thought Prompts.

SOTA Performance and Outlook: Across all tested models and datasets, RoGTR consistently achieves state-of-the-art (SOTA) performance. Moreover, its results are further improved with self-consistency ensembling. This not only validates the effectiveness of our proposed framework but also underscores that graph-based table representation remains a promising avenue for future exploration.

C. Ablation Study

Our ablation study (Table II) confirms that all RoGTR components contribute to its final 0.882 EM accuracy. The largest gains come from Joint Reasoning (JR) (+0.041) and Semantic Enhancement (SE) (+0.036), validating our hybrid reasoning and context enrichment approach. As shown in Figure 4, JR's effectiveness stems from its ability to mitigate the weaknesses of its constituent modes: Direct reasoning excels at simple retrieval but fails on complex calculations, whereas Chain-of-Thought (CoT) shows the opposite behavior. JR synergistically applies the appropriate mode for each task. Further gains are provided by the Self-Consistency (SC) ensemble (+0.024) and the foundational Retrieval (Ret) step (+0.005), collectively affirming RoGTR's synergistic design.

TABLE II
ABLATION STUDY USING GPT-4O ON HITAB.

Configuration	EM Accuracy
RoGTR w/o JR, SE, Ret	0.776 (↓ 0.106)
RoGTR w/o JR, SE	0.781 (↓ 0.101)
RoGTR w/o JR	0.817 (↓ 0.065)
RoGTR	0.858 (↓ 0.024)
RoGTR (w/ SC)	0.882

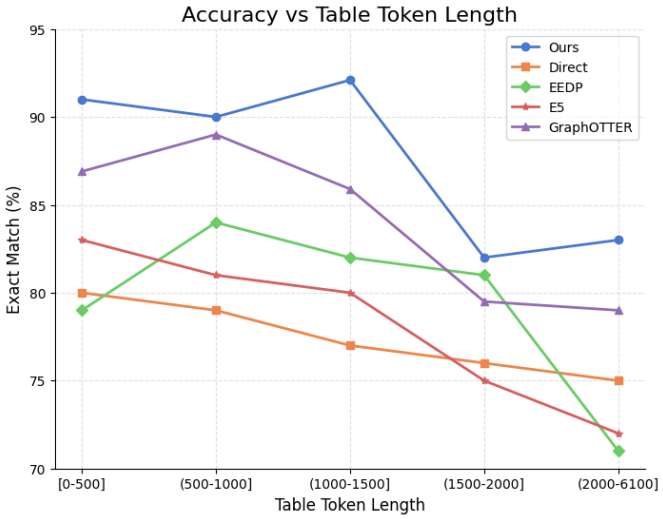


Fig. 5. Effect of Table Token Length on EM Accuracy.

D. Robustness Analysis

Robustness Against Complexity. To assess robustness across diverse table structures and question formulations, experiments were conducted on the WikiTQ benchmark (Table III). The results demonstrate that on the flat-table benchmark, our approach not only far surpasses methods designed for complex Table QA, but also outperforms many baselines specialized in flat table reasoning, thereby demonstrating strong robustness to variations in table structure.

TABLE III
METHODS MARKED WITH † USE A *GPT-3.5* BACKBONE; ALL OTHERS ARE BASED ON *GPT-4o-mini*.

Method	Exact Match (%)
Chain-of-table [22]	55.60
Binder [34]	58.86
Dater [20]	58.33
PoTable [35]	64.73
MIX-SC (DP) [11] †	66.99
E5 [33]	59.83
RoGTR	70.16
RoGTR w/ sc	**73.84**
SYNTQA (DP + Agent) [12]†	71.60
MIX-SC (DP + Agent) [11]†	73.06

Effect of Table Token Length. The impact of table token length on performance is analyzed in Figure 5. Performance of the **Direct** method generally degrades with increasing table token length. **EEDP** exhibits fluctuations but also trends downward with greater length. In contrast, our approach (**Ours**) maintains robust performance across varying lengths, demonstrating a significant advantage, especially for longer tables (2000-6100 tokens).

E. Efficiency Analysis

We analyze RoGTR's computational efficiency against the iterative GraphOTTER framework, focusing on two key metrics: Large Language Model (LLM) API calls per query and potential processing latency.

API Call Comparison. RoGTR employs a predefined workflow with a fixed base number of API calls. As detailed in Table IV, RoGTR's core process requires 3 LLM calls per query: one for Semantic Enhancement and two for parallel Zero-shot and Few-shot reasoning. An optional fourth call is invoked for self-reflection arbitration if initial answers diverge. Conversely, GraphOTTER's ReAct-style Thought-Action loop yields a variable number of API calls based on query complexity and LLM intermediate reasoning. GraphOTTER's observed average calls are slightly higher than RoGTR's base. This suggests RoGTR may offer lower, more predictable API usage overhead.

TABLE IV
COMPARISON OF LLM API CALLS AND TIME CONSUMPTION PER QUERY. †BASE CALLS: 1 FOR SEMANTIC ENHANCEMENT + 2 FOR PARALLEL REASONING PATHS (ZERO-SHOT & FEW-SHOT).

Method	API Calls per Query	Time (s)
RoGTR (Average)	3† (Fixed base)	10.2
GraphOTTER (Average)		
w/ `GPT-4o`	4.36	21.8
w/ `GPT-4o-mini`	4.14	16.5
E5 (Average)		
w/ `GPT-4o`	5.142	25.1
w/ `GPT-4o-mini`	8.213	30.4

Latency Considerations. Execution structure, beyond call numbers, significantly impacts overall latency. RoGTR leverages parallelism; its two primary reasoning pathways execute concurrently. Thus, the reasoning stage's effective latency is determined by the maximum of these parallel calls. Total latency is approximately the Semantic Enhancement call latency plus this maximum reasoning latency (and arbitration latency, if needed). Conversely, GraphOTTER's inherent Thought-Action paradigm is fundamentally sequential. Each thought and action requires a separate, sequential LLM invocation. Consequently, GraphOTTER's total processing time scales roughly linearly with API call count, reflecting cumulative latency of all reasoning steps. This sequential nature means RoGTR may achieve lower end-to-end latency with similar average API calls due to its parallelizable reasoning.

V. CONCLUSIONS

This work explores enhancing LLM-based tabular reasoning using *textual reasoning*. We introduce **RoGTR**, a lightweight method that serializes tables into a *graph of key information* via an intuitive, graph-based approach. Specifically designed for *complex, hierarchical tables*, RoGTR captures intricate semantic dependencies arising from multi-level row-column structures. Our work demonstrates that serializing tables into a graph structure provides a robust and effective foundation for subsequent tabular operations.

ACKNOWLEDGMENT

This work is funded by National Natural Science Foundation of China (No. NSFC62306276), Yongjiang Talent Introduction Programme (2022A-238-G), and Fundamental Research Funds for the Central Universities (226-2023-00138).

REFERENCES

[1] S.-J. Lim and Y.-K. Ng, "An automated approach for retrieving hierarchical data from html tables," in *Proceedings of the eighth international conference on Information and knowledge management*, 1999, pp. 466–474.

[2] Z. Wang, H. Dong, R. Jia, J. Li, Z. Fu, S. Han, and D. Zhang, "Structure-aware pre-training for table understanding with tree-based transformers," *arXiv:2010.12537*, 2020.

[3] X. Fang, W. Xu, F. A. Tan, J. Zhang, Z. Hu, Y. Qi, S. Nickleach, D. Socolinsky, S. Sengamedu, and C. Faloutsos, "Large language models(llms) on tabular data: Prediction, generation, and understanding – a survey," 2024. [Online]. Available: https://arxiv.org/abs/2402.17944

[4] W. Lu, J. Zhang, J. Fan, Z. Fu, Y. Chen, and X. Du, "Large language model for table processing: a survey," *Frontiers of Computer Science*, vol. 19, no. 2, Jan. 2025. [Online]. Available: http://dx.doi.org/10.1007/s11704-024-40763-6

[5] B. Zhao, C. Ji, Y. Zhang, W. He, Y. Wang, Q. Wang, R. Feng, and X. Zhang, "Large language models are complex table parsers," 2023. [Online]. Available: https://arxiv.org/abs/2312.11521

[6] J. Li, B. Hui, R. Cheng, B. Qin, C. Ma, N. Huo, F. Huang, W. Du, L. Si, and Y. Li, "Graphix-t5: Mixing pretrained transformers with graph-aware layers for text-to-sql parsing," in *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI)*, 2023.

[7] Y. Zhang, J. Henkel, A. Floratou, J. Cahoon, S. Deep, and J. M. Patel, "Reactable: Enhancing react for table question answering," 2023. [Online]. Available: https://arxiv.org/abs/2310.00815

[8] P. Li, Y. He, D. Yashar, W. Cui, S. Ge, H. Zhang, D. R. Fainman, D. Zhang, and S. Chaudhuri, "Table-gpt: Table-tuned gpt for diverse table tasks," 2023. [Online]. Available: https://arxiv.org/abs/2310.09263

[9] A. Su, A. Wang, C. Ye, C. Zhou, G. Zhang, G. Chen, G. Zhu, H. Wang, H. Xu, H. Chen, H. Li, H. Lan, J. Tian, J. Yuan, J. Zhao, J. Zhou, K. Shou, L. Zha, L. Long, L. Li, P. Wu, Q. Zhang, Q. Huang, S. Yang, T. Zhang, W. Ye, W. Zhu, X. Hu, X. Gu, X. Sun, X. Li, Y. Yang, and Z. Xiao, "Tablegpt2: A large multimodal model with tabular data integration," 2024. [Online]. Available: https://arxiv.org/abs/2411.02059

[10] X. He, Y. Liu, M. Zhou, Y. He, H. Dong, S. Han, Z. Yuan, and D. Zhang, "Tablelora: Low-rank adaptation on table structure understanding for large language models," 2025. [Online]. Available: https://arxiv.org/abs/2503.04396

[11] T. Liu, F. Wang, and M. Chen, "Rethinking tabular data understanding with large language models," 2023.

[12] S. Zhang, A. T. Luu, and C. Zhao, "SynTQA: Synergistic table-based question answering via mixture of text-to-SQL and E2E TQA," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 2352–2364. [Online]. Available: https://aclanthology.org/2024.findings-emnlp.131

[13] Y. Sui, M. Zhou, M. Zhou, S. Han, and D. Zhang, "Table meets llm: Can large language models understand structured table data? a benchmark and empirical study," 2024. [Online]. Available: https://arxiv.org/abs/2305.13062

[14] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, and J. Eisenschlos, "Tapas: Weakly supervised table parsing via pre-training," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020. [Online]. Available: http://dx.doi.org/10.18653/v1/2020.acl-main.398

[15] P. Yin, G. Neubig, W. tau Yih, and S. Riedel, "Tabert: Pretraining for joint understanding of textual and tabular data," 2020. [Online]. Available: https://arxiv.org/abs/2005.08314

[16] Q. Li, C. Huang, S. Li, Y. Xiang, D. Xiong, and W. Lei, "Graphotter: Evolving llm-based graph reasoning for complex table question answering," 2024. [Online]. Available: https://arxiv.org/abs/2412.01230

[17] T. Zhang, X. Yue, Y. Li, and H. Sun, "Tablellama: Towards open large generalist models for tables," 2024. [Online]. Available: https://arxiv.org/abs/2311.09206

[18] S. Chang and E. Fosler-Lussier, "Selective demonstrations for cross-domain text-to-sql," 2023. [Online]. Available: https://arxiv.org/abs/2310.06302

[19] D. Gao, H. Wang, Y. Li, X. Sun, Y. Qian, B. Ding, and J. Zhou, "Text-to-sql empowered by large language models: A benchmark evaluation," 2023. [Online]. Available: https://arxiv.org/abs/2308.15363

[20] Y. Ye, B. Hui, M. Yang, B. Li, F. Huang, and Y. Li, "Large language models are versatile decomposers: Decompose evidence and questions for table-based reasoning," 2023. [Online]. Available: https://arxiv.org/abs/2301.13808

[21] J. Jiang, K. Zhou, Z. Dong, K. Ye, W. X. Zhao, and J.-R. Wen, "Structgpt: A general framework for large language model to reason over structured data," 2023. [Online]. Available: https://arxiv.org/abs/2305.09645

[22] Z. Wang, H. Zhang, C.-L. Li, J. M. Eisenschlos, V. Perot, Z. Wang, L. Miculicich, Y. Fujii, J. Shang, C.-Y. Lee, and T. Pfister, "Chain-of-table: Evolving tables in the reasoning chain for table understanding," 2024. [Online]. Available: https://arxiv.org/abs/2401.04398

[23] P. Srivastava, M. Malik, V. Gupta, T. Ganu, and D. Roth, "Evaluating llms' mathematical reasoning in financial document question answering," 2024. [Online]. Available: https://arxiv.org/abs/2402.11194

[24] S. Saha, X. V. Yu, M. Bansal, R. Pasunuru, and A. Celikyilmaz, "Murmur: Modular multi-step reasoning for semi-structured data-to-text generation," 2022. [Online]. Available: https://arxiv.org/abs/2212.08607

[25] Z. Cheng, H. Dong, Z. Wang, R. Jia, J. Guo, Y. Gao, S. Han, J.-G. Lou, and D. Zhang, "Hitab: A hierarchical table dataset for question answering and natural language generation," *arXiv preprint arXiv:2108.06712*, 2021.

[26] Y. Katsis, S. Chemmengath, V. Kumar, S. Bharadwaj, M. Canim, M. Glass, A. Gliozzo, F. Pan, J. Sen, K. Sankaranarayanan, and S. Chakrabarti, "Ait-qa: Question answering dataset over complex tables in the airline industry," 2021.

[27] P. Pasupat and P. Liang, "Compositional semantic parsing on semi-structured tables," 2015. [Online]. Available: https://arxiv.org/abs/1508.00305

[28] OpenAI, "Gpt-4o mini: Advancing cost-efficient intelligence," 2024, accessed: 2025-04-24. [Online]. Available: https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/

[29] ——, "Gpt-4o system card," 2024, accessed: 2025-04-24. [Online]. Available: https://openai.com/index/gpt-4o-system-card/

[30] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang, G. Dong, H. Wei, H. Lin, J. Tang, J. Wang, J. Yang, J. Tu, J. Zhang, J. Ma, J. Xu, J. Zhou, J. Bai, J. He, J. Lin, K. Dang, K. Lu, K. Chen, K. Yang, M. Li, M. Xue, N. Ni, P. Zhang, P. Wang, R. Peng, R. Men, R. Gao, R. Lin, S. Wang, S. Bai, S. Tan, T. Zhu, T. Li, T. Liu, W. Ge, X. Deng, X. Zhou, X. Ren, X. Zhang, X. Wei, X. Ren, Y. Fan, Y. Yao, Y. Zhang, Y. Wan, Y. Chu, Y. Liu, Z. Cui, Z. Zhang, and Z. Fan, "Qwen2 technical report," *arXiv preprint arXiv:2407.10671*, 2024.

[31] M. Abdin, J. Aneja, H. Behl, S. Bubeck, R. Eldan, S. Gunasekar, M. Harrison, R. J. Hewett, M. Javaheripi, P. Kauffmann, J. R. Lee, Y. T. Lee, Y. Li, W. Liu, C. C. T. Mendes, A. Nguyen, E. Price, G. de Rosa, O. Saarikivi, A. Salim, S. Shah, X. Wang, R. Ward, Y. Wu, D. Yu, C. Zhang, and Y. Zhang, "Phi-4 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2412.08905

[32] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[33] Z. Zhang, Y. Gao, and J.-G. Lou, "e^5: Zero-shot hierarchical table analysis using augmented LLMs via explain, extract, execute, exhibit and extrapolate," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, K. Duh, H. Gomez, and S. Bethard, Eds. Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 1244–1258. [Online]. Available: https://aclanthology.org/2024.naacl-long.68/

[34] Z. Cheng, T. Xie, P. Shi, C. Li, R. Nadkarni, Y. Hu, C. Xiong, D. Radev, M. Ostendorf, L. Zettlemoyer, N. A. Smith, and T. Yu, "Binding language models in symbolic languages," 2023. [Online]. Available: https://arxiv.org/abs/2210.02875

[35] Q. Mao, Q. Liu, Z. Li, M. Cheng, Z. Zhang, and R. Li, "Potable: Towards systematic thinking via stage-oriented plan-then-execute reasoning on tables," 2025. [Online]. Available: https://arxiv.org/abs/2412.04272