RITT: A Retrieval-Assisted Framework with Image and Text Table Representations for Table Question Answering

Wei Zhou[1,3] **Mohsen Mesgar**[1] **Heike Adel**[2] **Annemarie Friedrich**[3]
[1]Bosch Center for Artificial Intelligence, Renningen, Germany
[2]Hochschule der Medien, Stuttgart, Germany [3]University of Augsburg, Germany
{wei.zhou|mohsen.mesgar}@de.bosch.com
annemarie.friedrich@uni-a.de adel-vu@hdm-stuttgart.de

Abstract

Tables can be represented either as text or as images. Previous works on table question answering (TQA) typically rely on only one representation, neglecting the potential benefits of combining both. In this work, we explore integrating textual and visual table representations using multi-modal large language models (MLLMs) for TQA. Specifically, we propose RITT, a retrieval-assisted framework that first identifies the most relevant part of a table for a given question, then dynamically selects the optimal table representations based on the question type. Experiments demonstrate that our framework significantly outperforms the baseline MLLMs by an average of 13 Exact Match and surpasses two text-only state-of-the-art TQA methods on four TQA benchmarks, highlighting the benefits of leveraging both textual and visual table representations.

1 Introduction

Previous approaches in table question answering (TQA) represent tables as either textual sequences (Herzig et al., 2020; Jiang et al., 2022; Zhang et al., 2023a) or as images (Zheng et al., 2024; Deng et al., 2024a), and process them by large language models (LLMs) or multi-modal large language models (MLLMs) accordingly. However, in real-life scenarios, tables often exist in both forms (e.g., HTML tables), or one form can be easily converted to the other via optical character recognition (OCR) or HTML rendering. This leads to increased interest in approaches that leverage both visual and textual table representations (Deng et al., 2024b; Liu et al., 2025; Zhou et al., 2025).

Current approaches using both representations either fine-tune an existing MLLM using preference data collected by prompting MLLMs with different table representations of a TQA problem (Liu et al., 2025), or leverage instance-level features (e.g., table size) to determine the best representation for an MLLM to process (Zhou et al., 2025).

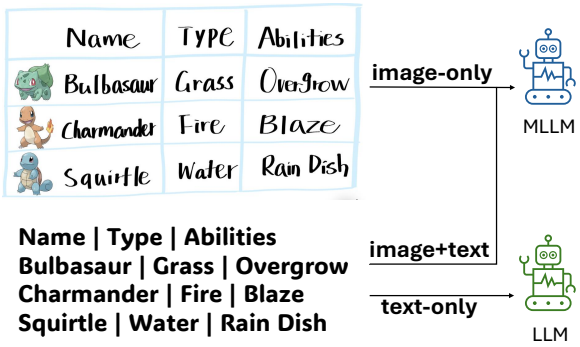

Figure 1: Three current approaches for representing and processing tables. Our framework RITT uses both table images and texts.

The former requires careful data collection and training (Feng et al., 2024), while the latter struggles to effectively handle large tables due to the inherent limitations of current MLLMs (Li et al., 2023; Zhou et al., 2025).

This work builds upon the latter approach, focusing on designing a training-free framework that can be easily applied across different datasets. We adopt the core idea proposed in FRES (Zhou et al., 2025) to select the most suitable table representation of a TQA problem based on its question type. Questions are classified as either retrieval questions, which only require locating information to be solved, or reasoning questions, which require both retrieval and reasoning.

However, unlike FRES, we introduce a novel **a sub-table retriever** that selects the most relevant part of a table to reduce input size. The module produces relevant table texts and images, which can be combined with the original full table and passed to an MLLM for reasoning. To determine the optimal representation combinations for an MLLM, we **extend the analysis** from Zhou et al. (2025) to explore combinatorial scenarios, such as pairing retrieved table images with original textual representations and vice versa. Our results indicate

Proceedings of the 4th Table Representation Learning Workshop, pages 86–97

that combining textual and visual representations yields the best performance for reasoning questions, while textual representation alone is sufficient for retrieval questions.

Based on these findings, we propose RITT, a training-free Retrieval-assisted framework leveraging Image and Text representations of Tables. It comprises four modules: a sub-table retriever, a question classifier, a table reformatter, and an MLLM reasoner. Experimental results show that RITT outperforms baseline MLLMs by an average of 13 exact match (EM) points, and surpasses two state-of-the-art text-only TQA systems, demonstrating the clear benefits of leveraging both table representations. Lastly, we provide an ablation study highlighting the contribution of each component.

2 Related Work

Sub-table Retrieval. Both LLMs and MLLMs have been shown to struggle with large tables (Lin et al., 2023; Wang et al., 2024a). To address this issue, prior work either fine-tunes smaller retriever models using annotated gold sub-tables (Lin et al., 2023; Lee et al., 2024), or utilizes LLMs as retrievers via in-context learning (Chen et al., 2024; Li et al., 2024b; Ye et al., 2023). In this work, we propose an LLM-based sub-table retriever. Unlike existing approaches that rely solely on semantic matching between headers and cells (Chen et al., 2024; Li et al., 2024b), our method further narrows down relevant cells by explicitly formulating and executing filtering logic. In contrast to methods that directly output relevant row indices using an LLM (Ye et al., 2023), our retriever ensures faithful generation using code execution for relevant content filtering.

Table Representations for TQA. Most prior work processes tables as texts (Zhang et al., 2023a; Wang et al., 2024c) or as images (Zheng et al., 2024). Liu et al. (2025) fine-tune an existing MLLM using preference data collected by prompting an MLLM with different table representations. Zhou et al. (2025) propose a rule-based framework FRES to select the best table representation for an MLLM. They obtain the rules by comparing different representations under varying scenarios controlled by table size and question type. Their findings indicate that different representations perform differently under varying conditions. For instance, passing large tables in textual format to LLMs can lead to better performance than passing large tables in images to MLLMs. Though the textual format is more robust than the visual format when handling larger tables, it still faces challenges with large tables. In this work, we propose a sub-table retriever to mitigate the impact of table size on representation selection. Moreover, we extend existing analyses to cover combinatorial cases where retrieved sub-tables are combined with original tables.

3 Framework

Figure 2 shows an overview of our proposed system RITT. It contains four parts: a *sub-table retriever* that filters for the most relevant cells, a *question classifier* to determine a question type, a *table reformatter* to reformat a retrieved sub-table based on question type, and a *table reasoner* MLLM to output an answer to a given TQA problem.

3.1 Sub-table Retrieval

We define a table T as a set of headers H, values V, and a schema function S that maps values to their corresponding headers. H can be further represented as $\{\emptyset, \{ht_1, \ldots, ht_m\}\} \cup \{\emptyset, \{hl_1, \ldots, hl_n\}\}$, where ht and hl stands for top and left headers, respectively, and m and n represent the number of columns and rows, respectively. We use $\emptyset$ to denote the absence of a header. For instance, the table in Figure 2 features only top headers. As a result, H can be represented as {"Country", "Result", "Year", "Score"}. The task of sub-table retrieval involves locating relevant top headers, ht_r, relevant left headers hl_r, and a set of cell values V_r indexed by those headers. As shown in the yellow box in Figure 2, an LLM takes in a TQA problem and can perform two tasks: header prediction and question parsing. Header prediction requires an LLM to predict the most relevant headers given a problem. Prompts for header prediction are shown in Figure 4. For a table that features both ht and hl, we directly aggregate cells indexed by predicted headers as a sub-table.

However, when either ht or hl is missing,[1] simply filtering based on available headers may still yield overly large sub-tables. To address this issue, we additionally ask the LLM to parse a question into filtering conditions in natural language (A prompt is shown in Figure 5). The same LLM then translates these conditions into executable Python code (A prompt is shown in Figure 6), which is run

[1]It is more common to have missing hl, e.g., relational tables, than missing ht.

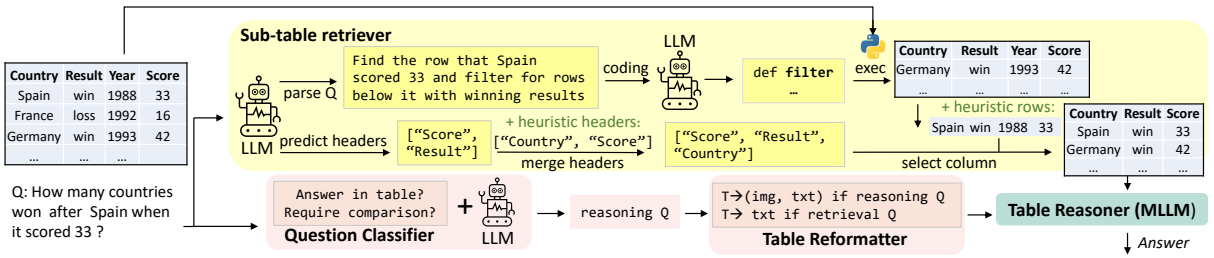

Figure 2: Given a table and question pair, a sub-table retriever outputs relevant cells. A question classifier is applied to distinguish retrieval and reasoning questions. Based on the question type, a table reformatter prepares an input table for a table reasoner, which outputs a final answer based on relevant cells and a question.

via a Python interpreter to further filter a table. If the code execution fails, we revert to the original table. Finally, we filter a retrieved sub-table based on relevant headers predicted by the LLM previously to produce a final sub-table.

To ensure the retrieved sub-table preserves essential information to solve a question, we also apply a heuristic: we include rows and columns that contain tokens in the question (referred to as heuristic rows/columns). For example, as shown in Figure 2, the heuristic headers "Country" and "Score" and heuristic rows mentioning "Spain" and "33" are added back to the final sub-table.

3.2 Question Classifier & Table Reformatter

Given a retrieved sub-table in textual format, the next step is to determine which table representations to pass to the table reasoner. Motivated by previous findings that MLLMs with table images manifest stronger reasoning abilities (Zhou et al., 2025), we consider determining table representations based on question type. Following Zhou et al. (2025), we classify questions into two categories: retrieval and reasoning. Retrieval questions are those whose answers can be directly located verbatim in the table cells, whereas reasoning questions require additional inference, involving numerical, temporal, or commonsense reasoning.

To investigate the effectiveness of table representations with different question types, we use the dataset provided by Zhou et al. (2025), which contains 1,600 instances from six common TQA datasets: WTQ (Pasupat and Liang, 2015), TabFact (Chen et al., 2020), HiTab (Cheng et al., 2022), CRT (Zhang et al., 2023b), TabMWP (Lu et al., 2022), and TempTabTQA (Gupta et al., 2023), with 800 instances for each question type. We examine multiple methods of passing retrieved sub-tables to the table reasoner: *rc*: passing only relevant headers in a prompt. *rt*: passing only relevant cells in a prompt. *ri*: passing relevant cells converted into image format. *rt+ri*: passing relevant cells in both text and image formats. Since providing only relevant cells may result in information loss, we also explore combining relevant cells with the original table: *ri+t*: passing relevant cells as images and the original table as text. *rt+i*: passing relevant cells as text and the original table as an image.

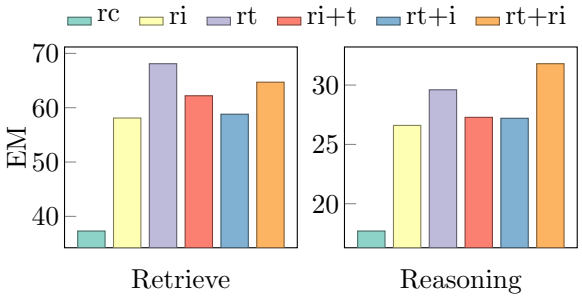

Figure 3: Comparing the effectiveness of different methods under varying question types. rc stands for passing relevant headers in a prompt. ri and rt represent passing only relevant table images and texts, respectively. ri+t stands for passing relevant cells as images and a full table as text. rt+i refers to passing relevant cells as texts and passing the original table as images. Lastly, rt+ti refers to passing both relevant cells as texts and images. We employ the Exact Match (EM) metric.

We evaluate these methods using six open-weight MLLMs as in Zhou et al. (2025): Qwen-2-VL-7b (Wang et al., 2024b), Pixtral-12b (Agrawal et al., 2024), Phi-3.5-vision-instruct-4b (Abdin et al., 2024), LLaVA-Next-7b (Li et al., 2024a), GLM-4v-9b (Zeng et al., 2024), and InternVL2-8b.[2] Exact Match is used for evaluation, which checks if a predicted answer and a ground truth are the same. To obtain relevant cells, we apply the method proposed in Section 3.1 on the evaluation dataset, with Qwen-2-72b as the backbone LLM.

[2] https://internvl.github.io/blog/2024-07-02-InternVL-2.0/

As Figure 3 shows, different question types benefit from different representations. For reasoning questions, passing relevant cells simultaneously as texts and images (rt+ri) to MLLMs achieves the best performance. In contrast, table text representation suffices when a question is of type retrieval. The observations align with findings in (Zhou et al., 2025). We do not observe a clear advantage from combining relevant table information with the original table information. We suspect this might be because adding the original table information back increases the input size. Detailed results for each MLLM are reported in Appendix A.3. Based on these observations, our table reformatter encodes sub-tables as both images and texts when a question is of type reasoning. Otherwise, only table texts are passed to the final reasoner. We adopt the question type classifier proposed by Zhou et al. (2024), which combines rule-based heuristics and an LLM. Details are represented in Appendix A.2.

4 Experiments

Datasets. We evaluate RITT with the **test sets** of three aforementioned TQA benchmarks used during our analysis in Section 3.2: WTQ, TabFact (small test), and HiTab. We also include one additional dataset, WikiSQL (Zhong et al., 2017), that has not been used in our analysis to test the generalizability of our method.

Models and Baselines. Our framework contains an MLLM table reasoner and an LLM retriever/classifier.[3] For MLLMs, we choose the best performing MLLM from our previous analysis: **Pixtral-12b** (see A.3 for individual model's performance) as well as a fine-tuned MLLM for TQA: **TableLlaVA-7b** (Zheng et al., 2024). We use Qwen-2-72b as the LLM backbone in our framework. As baselines, we choose the backbone MLLM without applying RITT. In addition, we compare RITT with two SoTA frameworks that use only table text representations: TableRAG (Chen et al., 2024) and GraphOTTER (Li et al., 2024b). Both frameworks involve relevant cell retrieval and are inference-based methods utilizing LLMs. For fair comparisons, we replace the LLM backbones used in previous work and this work with an MLLM (Qwen-2-VL-72b).[4] As a result, all compared frameworks use the same backbone model.

[3]The retriever/classifier can be replaced by the MLLM reasoner. We do not find big performance differences between an LLM and an MLLM of the same size and series.

Systems	WTQ	TabFact	HiTab	WiKiSQL
Pixtral-12b	52.5	75.9	62.2	60.1
+RITT	54.4 (+1.9)	76.8 (+0.9)	68.0 (+5.8)	62.7 (+2.6)
TableLlaVA-7b	17.2	60.9	16.3	29.5
+RITT	39.7(+22.5)	64.5 (+3.6)	61.8 (+45.5)	52.0 (+22.5)
Qwen2-VL-72b	62.6	**86.7**	73.4	77.6
+RITT	**63.4**	86.0	**76.4**	**78.0**
TableRAG	60.8	79.3	65.3	77.9
GraphOTTER	59.4	81.8	71.6	75.6

Table 1: Model performances on four TQA benchmarks. The first four rows show the results of direct inference with MLLMs and applying our framework. The last four rows compare the performance of our framework with two SoTA systems using the same model.

5 Results and Discussions

Results. Table 1 shows the results averaged across three runs. The top section of the table compares direct inference using the backbone MLLMs against the same models enhanced by RITT. The bottom section compares our method to two state-of-the-art frameworks (TableRAG and GraphOTTER) using the same underlying backbone model. Applying RITT consistently improves performance, with notable gains on the HiTab dataset, achieving increases of 5.8 and 45.5 EM points for Pixtral-12b and TableLLaVA-7b, respectively. Differences in improvement can be attributed to the capabilities of the base models: Pixtral-12b generally exhibits stronger multi-modal reasoning capabilities than TableLlaVA and can handle longer inputs, resulting in smaller performance improvements. Moreover, our framework consistently outperforms both TableRAG and GraphOTTER across all four datasets. When using a larger MLLM model (Qwen-2-72b), we still observe improvements in three out of four datasets, though these improvements are relatively smaller compared to those observed with smaller models. We hypothesize this is because larger models inherently have stronger capabilities in handling longer and more complex table-question instances, leaving less room for improvement from additional retrieval steps.

Ablation. We conduct an ablation study to analyze the contribution of each component. We present results categorized by table size to understand how each component performs on tables of

[4]We do not find a significant performance difference when switching from a text-only model to a multi-modal model.

System #Instances	0-50 187	50-100 413	100-200 466	>200 518
MLLM	74.3	63.7	61.6	57.3
+ st	+4.1	+1.6	+2.9	+7.5
+ st+tr	+3.8	+2.9	+4.3	+10.3
+ st+tr (oracle)	+4.3	+3.1	+5.1	+11.7

Table 2: Ablation study of our framework on HiTab using Pixtral-12b. st stands for sub-table retriever, and tr stands for table reformatter. tr (oracle) refers to passing the oracle question type obtained from the dataset.

Methods	Precision	Recall	F_1	# Cells
GraphOTTER	**47.4**	51.1	46.8	4.56
TableRAG	17.6	40.3	22.7	13.8
RITT	41.0	**92.6**	**51.2**	8.65

Table 3: Comparing our sub-table retriever with two state-of-the-art sub-table retrievers.#Cells shows the average number of identified relevant cells. For gold relevant cells, the number is 5.34.

varying sizes. When ablating the table reformatter, retrieved sub-tables are passed to an MLLM as both images and texts. We pass the oracle question types obtained from the dataset annotation to investigate the effectiveness of the question classifier. Table 2 shows results on the HiTab dataset, chosen as it exhibits the largest performance gains using RITT. We observe that both the sub-table retriever and table reformatter contribute to the overall performance. The sub-table retriever demonstrates greater performance enhancement compared to the table reformatter. Additionally, we note that overall system performance tends to decline as the table size increases, aligning with previous findings (Lin et al., 2023). Interestingly, the benefit provided by the sub-table retriever becomes more pronounced on larger tables, highlighting its effectiveness in handling large tables.

Effectiveness of Sub-table Retriever. We compare our proposed sub-table retriever with two current state-of-the-art LLM-based retrievers (introduced in Section 4) on 800 instances from the HiTab evaluation subset described in Section 3.2. We chose this dataset as it provides manual annotations of relevant table cells required to answer each question. The results are shown in Table 3. Our proposed sub-table retriever achieves the highest F_1 score among the three methods, demonstrating its effectiveness in accurately identifying relevant table cells. Nevertheless, the high recall and relatively large average number of cells (8.65 compared to the gold standard of 5.34) indicate that our sub-table retriever identified irrelevant cells with regard to answering a question.

Error Analysis. We randomly sample 100 instances on which applying RITT with Pixtral-12b fails, with each investigated dataset 25 instances, and perform an error analysis. For each instance, we manually check (1) whether a retrieved sub-table contains the relevant information needed to answer a question, and (2) whether the question type is predicted correctly. We find that for approximately 23% of instances, the retrieved sub-tables do not contain the information needed to answer questions, leading to information loss. In contrast, only 7% of instances are predicted with wrong question types, suggesting the task is relatively easy. We observe that in the majority of cases, the relevant information is present in the retrieved sub-tables, and the question type is correctly identified. However, Pixtral-12b still fails to provide the correct answer. This might be because the retrieved sub-tables are still large, due to code execution errors during row filtering. The failure affects instances with reasoning questions more than retrieval questions, given that both table images and table texts are passed when a question is of type reasoning. An example is provided in Figure 7. Our analysis suggests that future work should focus on developing methods that reduce table sizes effectively without losing necessary information.

6 Conclusions

In this paper, we explored leveraging both textual and visual table representations using MLLMs for TQA. To handle the challenges of large table inputs and representation selection, we proposed RITT, a retrieval-assisted framework that retrieves the most relevant sub-table, classifies the question type, and dynamically determines the optimal representations to an MLLM reasoner based on the question type. Extensive experiments on four TQA benchmarks demonstrated the advantages of our framework over baseline MLLMs as well as frameworks utilizing only textual representations. Ablation studies further confirmed the effectiveness of each proposed component. Our findings highlight the benefits and promising potential of integrating both table representations for TQA.

Limitation

We explore utilizing both textual and visual table representations. Nevertheless, the underlying assumption that table images and table texts both exist and can be easily converted might not hold for every case. For instance, converting large table images to texts using OCR tools can suffer from information loss. We leave these for further exploration. Secondly, due to a limited number of existing large MLLMs, we specifically focus on evaluating and designing methods for small MLLMs. Last but not least, RITT is a pipeline method, consisting of several components. As a result, it requires longer inference time than end-to-end systems.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Hassan Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Singh Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Caio C'esar Teodoro Mendes, Weizhu Chen, Vishrav Chaudhary, Parul Chopra, Allison Del Giorno, Gustavo de Rosa, Matthew Dixon, Ronen Eldan, Dan Iter, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Jamie Huynh, Mojan Javaheripi, Xin Jin, Piero Kauffmann, Nikos Karampatziakis, Dongwoo Kim, Young Jin Kim, Mahoud Khademi, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Chen Liang, Weishung Liu, Eric Lin, Zeqi Lin, Piyush Madan, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Corby Rosset, Sambudha Roy, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Xianmin Song, Olatunji Ruwase, Praneetha Vaddamanu, Xin Wang, Rachel Ward, Guanhua Wang, Philipp Witte, Michael Wyatt, Can Xu, Jiahang Xu, Sonali Yadav, Fan Yang, Ziyi Yang, Donghan Yu, Cheng-Yuan Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yunan Zhang, and Xiren Zhou. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *ArXiv*, abs/2404.14219.

Pravesh Agrawal, Szymon Antoniak, Emma Bou Hanna, Devendra Singh Chaplot, Jessica Chudnovsky, Saurabh Garg, Théophile Gervet, Soham Ghosh, Am'elie H'eliou, Paul Jacob, Albert Q. Jiang, Timothée Lacroix, Guillaume Lample, Diego de Las Casas, Thibaut Lavril, Teven Le Scao, Andy Lo, William Marshall, Louis Martin, Arthur Mensch, Pavankumar Reddy Muddireddy, Valera Nemychnikova, Marie Pellat, Patrick von Platen, Nikhil Raghuraman, Baptiste Rozière, Alexandre Sablayrolles, Lucile Saulnier, Romain Sauvestre, Wendy Shang, Roman Soletskyi, Lawrence Stewart, Pierre Stock, Joachim Studnia, Sandeep Subramanian, Sagar Vaze, and Thomas Wang. 2024. Pixtral 12b. *ArXiv*, abs/2410.07073.

Si-An Chen, Lesly Miculicich, Julian Martin Eisenschlos, Zifeng Wang, Zilong Wang, Yanfei Chen, Yasuhisa Fujii, Hsuan-Tien Lin, Chen-Yu Lee, and Tomas Pfister. 2024. Tablerag: Million-token table understanding with language models. *ArXiv*, abs/2410.04739.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. Tabfact : A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2022. HiTab: A hierarchical table dataset for question answering and natural language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1094–1110, Dublin, Ireland. Association for Computational Linguistics.

Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. 2024a. Investigating data contamination in modern benchmarks for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8706–8719, Mexico City, Mexico. Association for Computational Linguistics.

Naihao Deng, Zhenjie Sun, Ruiqi He, Aman Sikka, Yulong Chen, Lin Ma, Yue Zhang, and Rada Mihalcea. 2024b. Tables as texts or images: Evaluating the table reasoning ability of LLMs and MLLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 407–426, Bangkok, Thailand. Association for Computational Linguistics.

Duanyu Feng, Bowen Qin, Chen Huang, Zheng Zhang, and Wenqiang Lei. 2024. Towards analyzing and understanding the limitations of dpo: A theoretical perspective.

Vivek Gupta, Pranshu Kandoi, Mahek Vora, Shuo Zhang, Yujie He, Ridho Reinanda, and Vivek Srikumar. 2023. TempTabQA: Temporal question answering for semi-structured tables. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2431–2453, Singapore. Association for Computational Linguistics.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Zhengbao Jiang, Yi Mao, Pengcheng He, Graham Neubig, and Weizhu Chen. 2022. OmniTab: Pretraining with natural and synthetic data for few-shot table-based question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 932–942, Seattle, United States. Association for Computational Linguistics.

Wonjin Lee, Kyumin Kim, Sungjae Lee, Jihun Lee, and Kwang In KIm. 2024. Piece of table: A divide-and-conquer approach for selecting sub-tables in table question answering.

Feng Li, Renrui Zhang, Hao Zhang, Yuanhan Zhang, Bo Li, Wei Li, Zejun Ma, and Chunyuan Li. 2024a. Llava-next-interleave: Tackling multi-image, video, and 3d in large multimodal models. *ArXiv*, abs/2407.07895.

Qianlong Li, Chen Huang, Shuai Li, Yuanxin Xiang, Deng Xiong, and Wenqiang Lei. 2024b. Graphotter: Evolving llm-based graph reasoning for complex table question answering.

Zhang Li, Biao Yang, Qiang Liu, Zhiyin Ma, Shuo Zhang, Jingxu Yang, Yabo Sun, Yuliang Liu, and Xiang Bai. 2023. Monkey: Image resolution and text label are important things for large multi-modal models. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26753–26763.

Weizhe Lin, Rexhina Blloshmi, Bill Byrne, Adria de Gispert, and Gonzalo Iglesias. 2023. An inner table retriever for robust table question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9909–9926, Toronto, Canada. Association for Computational Linguistics.

Zhenghao Liu, Haolan Wang, Xinze Li, Qiushi Xiong, Xiaocui Yang, Yu Gu, Yukun Yan, Qi Shi, Fangfang Li, Ge Yu, and Maosong Sun. 2025. Hippo: Enhancing the table understanding capability of large language models through hybrid-modal preference optimization.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and A. Kalyan. 2022. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. *ArXiv*, abs/2209.14610.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Hengyi Wang, Haizhou Shi, Shiwei Tan, Weiyi Qin, Wenyuan Wang, Tunyu Zhang, Akshay Uttama Nambi, Tanuja Ganu, and Hao Wang. 2024a. Multimodal needle in a haystack: Benchmarking long-context capability of multimodal large language models. *ArXiv*, abs/2406.11230.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Ke-Yang Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024b. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *ArXiv*, abs/2409.12191.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2024c. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *ArXiv*, abs/2401.04398.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 174–184, New York, NY, USA. Association for Computing Machinery.

Team Glm Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Ming yue Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiaoyu Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yi An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhenyi Yang, Zhengxiao Du, Zhen-Ping Hou, and Zihan Wang. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *ArXiv*, abs/2406.12793.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2023a. Tablellama: Towards open large generalist models for tables. In *North American Chapter of the Association for Computational Linguistics*.

Zhehao Zhang, Xitao Li, Yan Gao, and Jian-Guang Lou. 2023b. CRT-QA: A dataset of complex reasoning question answering over tabular data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2131–2153, Singapore. Association for Computational Linguistics.

Mingyu Zheng, Xinwei Feng, Qingyi Si, Qiaoqiao She, Zheng Lin, Wenbin Jiang, and Weiping Wang. 2024. Multimodal table understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9102–9124, Bangkok, Thailand. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Wei Zhou, Mohsen Mesgar, Heike Adel, and Annemarie Friedrich. 2024. FREB-TQA: A fine-grained robustness evaluation benchmark for table question answering. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2479–2497, Mexico City, Mexico. Association for Computational Linguistics.

Wei Zhou, Mohsen Mesgar, Heike Adel, and Annemarie Friedrich. 2025. Texts or images? a fine-grained analysis on the effectiveness of input representations and models for table question answering.

A Appendix

A.1 Prompts

We present prompts used in sub-table retriever in Figure 4, 5 and 6.

A.2 Datasets

Question Type Classification. We use the question type classifier proposed in Zhou et al. (2024): a rule-based method is applied first. If an answer is not in a table, a question is classified as a reasoning question. If a question contains comparative terms (detected using NLTK), the question is classified into a reasoning question. Next, an LLM takes in a question and table and returns a predicted question type. We replace the LlaMA-2-13b used in the original paper with Qwen-2-72b for its better general capabilities but keep the prompt the same.

Dataset Licenses WTQ (Pasupat and Liang, 2015), TabFact (Chen et al., 2020), HiTab (Cheng et al., 2022) and WikiSQL (Zhong et al., 2017), they are under the license of CC-BY-SA-4.0[5], MIT, BSD-3 CLAUSE[6] and C-UDA[7] respectively.

[5] https://creativecommons.org/licenses/by-sa/4.0/

[6] https://opensource.org/license/bsd-3-clause

[7] https://github.com/microsoft/HiTab?tab=License-1-ov-file

A.3 MLLMs

Table 4 shows performances of individual MLLMs on the evaluation set. We find that Pixtral 12b performs the best among all evaluated small models.

Header prediction prompt for hierarchical tables:
Your task is to find out the relevant headers based on the question.
Return the answer in json format: {'top_header':[(relevant top header tuple),...], 'left_header':[(relevant left header tuple),...]}
Below is an example:
top header: [('club',),('season',),('league','division'),('league','apps'),('league', 'goals'), ('total','apps'), ('total','goals')]
left header: [('Gillingham',),('Stevenage',),('Bristol City',)]
question: How many goals did this player score in total for Bristol City and Stevenage in League One?
answer: {'top_header':[('club,'), ('league', 'division'), ('league','goals'), ('total', 'goals')], 'left_header':[('Stevenage',),('Bristol City',)]}
now find out the relevant headers for the following instance:
top header: {top_header}
left header: {left_header}
question: {question}
answer:

Header prediction prompt for flat tables:
Your task is to find out the relevant headers to answer the question.
Return the answer in list format: ["relevant_header_a", "relevant_header_b",...] and nothing else.
Below is an example:
table: | country | result | year | score |
Example Row 1: | Spain | win | 2000 | 33 |
Example Row 2: | Germany | win | 2001 | 17 |
question: What is the next country to win after Germany?
Answer: ["country", "result", "year"]
now find the relevant headers for the following instance:
table: {table}
question: {question}

Figure 4: Prompts for header selection.

Model	QT	rc	rt+i	rt	ri+t	ri	rt+ri
Qwen2 7b	Retrieve	50.7	57.2	**81.2**	79.8	75.9	79.7
	Reasoning	24.4	29.1	37.0	35.8	38.2	**42.0**
Pixtral 12b	Retrieve	57.1	**77.9**	76.5	71.4	68.4	72.7
	Reasoning	32.2	40.1	39.7	41.1	39.9	**41.6**
Phi-3.5 4b	Retrieve	52.8	71.9	**77.4**	74.9	69.8	76.1
	Reasoning	17.4	28.3	28.2	27.3	24.7	**30.9**
LlaVA 7b	Retrieve	10.8	48.3	**68.5**	51.9	49.3	67.7
	Reasoning	3.85	12.0	20.4	12.9	9.9	**23.2**
GLM-4 9b	Retrieve	22.8	26.8	**33.0**	22.1	22.3	23.5
	Reasoning	10.5	12.2	12.6	11.1	13.3	**13.9**
Intern-8b	Retrieve	29.8	71.1	72.4	**73.2**	63.1	68.9
	Reasoning	18.3	37.7	**40.2**	34.7	34.1	39.5
Average	Retrieve	37.3	58.8	**68.1**	62.2	58.1	64.7
	Reasoning	17.7	26.5	29.6	27.2	26.6	**31.8**

Table 4: Exact Match of different MLLMs. QT stands for question type. rc refer to passing relevant column names in a prompt. ri and rt represent passing only relevant table images and texts, respectively. ri+t stands for passing relevant cells as images and full table as texts. rt+i refers to passing relevant cells as texts and passing original table as images. rt+ti refers to passing both relevant cells as texts and images. We employ the Exact Match (EM) metric.

Prompt for generating filtering conditions:

You are skilled at translating questions into filtering conditions and adhering to instructions. Your task is to convert a question into a dictionary containing filtering conditions and relevant columns. The dictionary should be structured as: {"general filtering statement: specific instructions for filtering":[list of relevant headers]}. Adhere strictly to this format. Use only words from the table's header when listing relevant columns. Only a portion of the table is shown for data type reference. Do not provide an answer to the question. Examples are provided below for clarity:

Example 1:
Table Header: | country | result | year |
Example Row 1: | Spain | win | 2000 |
Example Row 2: | Germany | win | 2001 |
Question: How many times did Spain win after 2001?
Answer: The question requires filtering for occurrences where Spain won after 2001. This involves checking rows where "country" is "Spain", "result" is "win", and "year" is after 2001. Relevant columns are ["country", "result", "year"]. Hence, the dictionary is: {"filter for rows where Spain won after 2001: find rows where country is Spain, result is win, and year is after 2001":["country", "result", "year"]}

Example 2:
Table Header: | country | result | year |
Example Row 1: | Spain | win | 2000 |
Example Row 2: | Germany | win | 2001 |
Question: What is the next country to win after Germany?
Answer: This question seeks the next winning country after Germany. It requires identifying when Germany won, then filtering for rows where "year" is greater than that year and "result" is "win". Relevant columns are ["country", "result", "year"]. The dictionary is: {"filter for rows where a win occurred after Germany: first identify the year Germany won, then find rows where year is later and result is win":["country", "result", "year"]}

Example 3:
Table Header: | team | scores |
Example Row 1: | Navi | 3 |
Example Row 2: | Spirit | 5 |
Question: What is the total score for Navis and G2?
Answer: This question asks for the total score of Navi and G2, requiring filters for rows where the team is either 'Navi' or 'G2'. Relevant column is ["team"]. The dictionary is: {"filter for rows where team is either Navis or G2: find rows where team is either Navi or G2":["team"]}

Now, based on the given table and question, compose the filtering conditions:
Table: {table}
Question: {question}
Answer:

Figure 5: Prompts for generating filtering conditions.

Prompt for code parsing:

Your task is to write a function 'filtering' to filter out irrelevant rows from a dataframe object, based on a given condition.
The given condition might not match the values or datatype in the dataframe. Therefore, you will have to translate the given condition into the dataframe operatable Python code or converting the data (type) in the dataframe. Return the filtered df in the variable 'df_filtered'
Below is an example:

df = pd.DataFrame.from_dict({{'country':['Spain', 'Germany', 'France', 'Norway'], 'year':['2000', '2001', '2002','1998']}})
condition = 'filter for rows that won after Germany: first find the year Germany won. Then filter for rows where year is later than the year Germany won'

answer: The condition selects rows where the 'year' should be larger than (after) the year when Germany won. We have to first find out when Germany won, and then filtering for rows that satisfy the condition. The corresponding Python code is:

```
def filtering(df):
# convert data type
  df['year'].astype('int64')
# find out the year when Germany won
  germany_won_year = df[df['country']=='Germany][['year'].tolist()[0]
# filter the table for rows that won after Germany won year
  df_filtered = df[df['year']>germany_won_year]
return df_filtered
```

Now please think carefully and write Python code to select relevant rows for the following dataframe based on the condition.
df = pd.DataFrame.from_dict({df_dict})
condition = {cond}
answer:

Figure 6: Prompts for code generation.

Question: What is the value Others% when the value Others# is greater than 147 and the value Kerry% is 39.6%?

Question type: reasoning

Original table

County	Kerry%	Kerry#	Bush%	Bush#	Others%	Others#
Adams	52.1%	5,447	46.8%	4,890	1.1%	119
...remaining 32 rows not shown...						
Sheboygan	44.1%	27,608	55.0%	34,458	0.9%	559

Sub-table

Kerry%	Kerry#	Others%	Others#
52.1%	5,447	1.1%	119
...remaining 32 rows not shown...			
44.1%	27,608	0.9%	559

image + text

Pixtral (12b)

Predicted answer: 1.3%

Gold answer: 1.1%

Figure 7: An error case of large sub-tables.