When TableQA Meets Noise: A Dual Denoising Framework for Complex Questions and Large-scale Tables

Shenghao Ye[1]*, **Yu Guo**[1]*, **Dong Jin**[3]†, **Yuxiang Wang**[2], **Yikai Shen**[1], **Yunpeng Hou**[3], **Shuangwu Chen**[1]†
Jian Yang[1], **Xiaofeng Jiang**[1]
[1]University of Science and Technology of China [2] The University of Melbourne
[3]Institute of Artificial Intelligence, Hefei Comprehensive National Science Center
{ssh0321y, yukariguo, shenyikai, hyp314}@mail.ustc.edu.cn
{kingdon, chensw, jianyang, jxf}@ustc.edu.cn

Abstract

Table question answering (TableQA) is a fundamental task in natural language processing (NLP). The strong reasoning capabilities of large language models (LLMs) have brought significant advances in this field. However, as real-world applications involve increasingly complex questions and larger tables, substantial noisy data is introduced, which severely degrades reasoning performance. To address this challenge, we focus on improving two core capabilities: Relevance Filtering, which identifies and retains information truly relevant to reasoning, and Table Pruning, which reduces table size while preserving essential content. Based on these principles, we propose EnoTab, a dual denoising framework for complex questions and large-scale tables. Specifically, we first perform Evidence-based Question Denoising by decomposing the question into minimal semantic units and filtering out those irrelevant to answer reasoning based on consistency and usability criteria. Then, we propose Evidence Tree-guided Table Denoising, which constructs an explicit and transparent table pruning path to remove irrelevant data step by step. At each pruning step, we observe the intermediate state of the table and apply a post-order node rollback mechanism to handle abnormal table states, ultimately producing a highly reliable sub-table for final answer reasoning. Finally, extensive experiments show that EnoTab achieves outstanding performance on TableQA tasks with complex questions and large-scale tables, confirming its effectiveness.

1 Introduction

Table question answering (TableQA) is a fundamental task in natural language processing (NLP) (Wang et al., 2022a) that focuses on answering questions based on tabular data. With the rapid development of large language models (LLMs) (OpenAI et al., 2024; Grattafiori et al., 2024; Team et al., 2024), this area has witnessed significant progress. However, in real-world applications such as healthcare and finance, both the scale of tables and the complexity of questions are increasing (Cafarella et al., 2008; Zhang et al., 2025), introducing substantial amounts of noisy data. Extensive research has demonstrated that such noisy data significantly degrades the performance of table reasoning (Chen et al., 2024; Wang et al., 2025a).

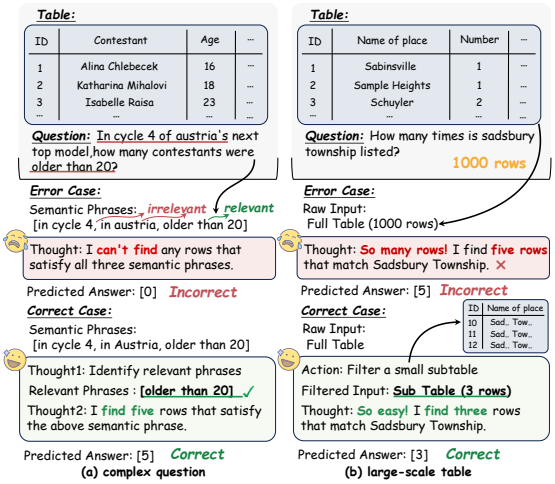

Figure 1: Error and Correct Cases for (a) the Complex Question and (b) the Large-Scale Table.

Where noisy data Comes From and Why it Matters? Noisy data in TableQA arises from both the question and the table (Zhou et al., 2025a; Wang et al., 2025a). Questions often contain spurious correlations. As shown in Figure 1(a), phrases such as "in cycle 4" and "in Austria" are mistakenly treated by the model as constraints, though no corresponding data exists in the table, ultimately leading to an incorrect answer. Tables, meanwhile, often contain irrelevant data. As shown in Figure 1(b), only 3 of 260 rows contribute to answering the question, while the remaining 257 are entirely irrel-

*Equal contribution
†Corresponding authors

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 24022–24045
July 2-7, 2026

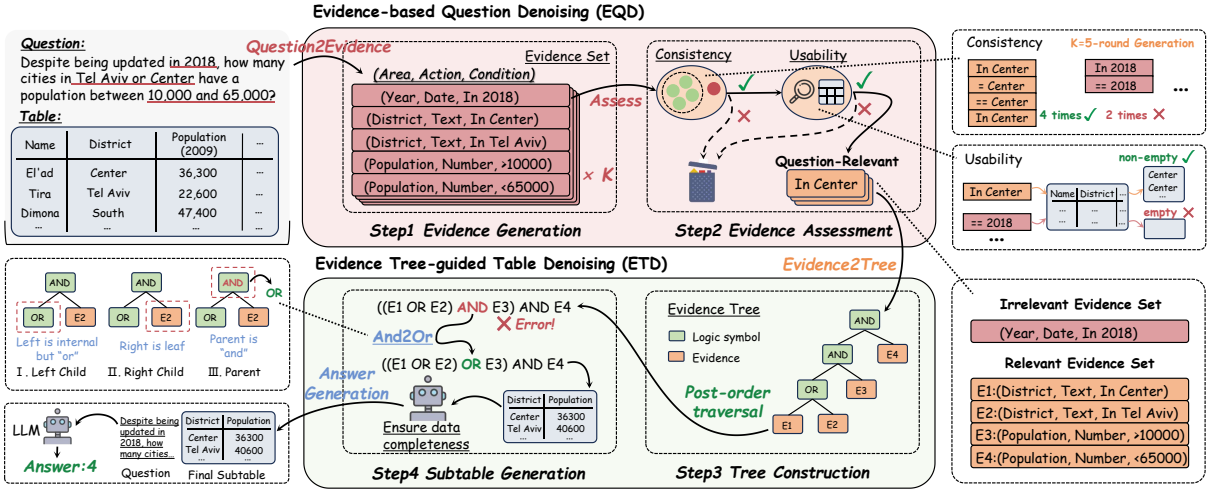

Figure 2: The EnoTab framework, composed of Evidence-based Question Denoising to remove irrelevant semantic units from the question and Evidence Tree-guided Table Denoising to eliminate irrelevant cell values from the table.

tational cost. In scenarios with complex questions and large-scale tables, noisy data becomes more pervasive and has greater impact on reasoning.

So, How to deal with noisy data? We observe that effective TableQA under substantial noisy data requires two indispensable capabilities. (1) **Relevance Filtering**, i.e., identifying and ignoring spurious correlations in the question (e.g., "in cycle 4" and "in Austria"). As shown in Figure 1(a), once the model is explicitly informed which phrases are irrelevant and which relevant, it can easily derive the correct answer. (2) **Table Pruning**, i.e., removing irrelevant data to reduce the table size. As shown in Figure 1(b), when the original 260-row table is reduced to a 3-row subtable by retaining only the question-relevant data, the model can infer the correct answer more efficiently and accurately. These two capabilities effectively mitigate the impact of noisy data on table reasoning performance.

But why do existing methods fail? Reviewing existing TableQA methods, we summarize their limitations into the following two aspects. First, it is hard to accurately identify spurious correlations in complex questions. Some existing methods prompt the LLM to directly eliminate spurious correlations in complex questions in order to derive simpler subquestions (Ye et al., 2023; Wang et al., 2024; Zhao et al., 2024), but this process is often error-prone, as the LLM tends to be overconfident and may mistakenly treat irrelevant elements as relevant. Other promising methods represent the question as a program and detect spurious correlations by checking whether the program executes correctly (Nahid and Rafiei, 2024b; Wang et al., 2025a). However, these correlations are often entangled with useful information, making it difficult to precisely locate and remove them even when their presence is detected. Secondly, it is easy to mistakenly lose target data relevant to answer reasoning when pruning the table. Most methods prune the table by generating SQL or Python programs (Zhang et al., 2023; Abhyankar et al., 2025). However, these methods often operate as black boxes, lacking transparency and making it hard to detect errors in time. Even if the loss of target data is identified, the pruning often has to be redone entirely (Wang et al., 2025a), resulting in increased overhead. In summary, two key challenges remain to be addressed: (1) how to accurately distinguish spurious correlations from truly relevant information in complex questions; and (2) how to effectively prune tables while preserving critical answer-related data.

To overcome these challenges, we propose EnoTab, which consists of two components: (1) **Evidence-based Question Denoising**, which introduces the concept of *Evidence* to decompose a question into multiple minimal semantic units. Each unit is assessed independently based on two criteria: whether it appears consistently across multi-turn generation and whether corresponding data can be found in the table, in order to determine its relevance to answer reasoning. (2) **Evidence Tree-guided Table Denoising**, which constructs an *Evidence Tree*, an explicit and transparent reasoning path for progressively pruning the table. Each pruning step is observable, allowing for timely detection of abnormal table states. To handle such abnormalities, we introduce a post-order node rollback mech-

anism to prevent the loss of answer-related data, ensuring that the final subtable remains reliable for supporting answer generation.

Our Contributions. (1) *New Perspective*. We reveal and analyze the performance bottleneck of TableQA when handling complex questions and large-scale tables with substantial noisy data, and attribute it primarily to insufficient capabilities in **Relevance Filtering** and **Table Pruning**. (2) *Novel Framework*. We propose EnoTab, a dual denoising framework designed to enhance noise robustness in TableQA tasks involving complex questions and large-scale tables. (3) *SOTA Performance*. Extensive experimental results demonstrate the effectiveness of EnoTab, which achieves outstanding performance in TableQA tasks involving complex questions and large-scale tables.

2 EnoTab

2.1 Problem Definition

TableQA consists of a question Q, a table T, and the corresponding answer Y. The question Q (e.g., "update in 2018, how many cities in Tel Aviv...?") can be represented as a set of semantic units s_i (e.g., "in 2018", "cities in Tel Aviv"), with $Q = \{s_1, s_2, ..., s_n\}$. The table T is represented as a two-dimensional matrix $T = \{c_{i,j} \mid 0 \leq i < N,\ 0 \leq j < M\}$, where N is the number of rows, M is the number of columns, and $c_{i,j}$ denotes the cell value at row i and column j. The answer $Y = (y_1, y_2, ..., y_n)$. The goal is to infer the correct answer Y given the question Q and the table T:

$$Y = \arg\max \prod_{i=1}^{n} P_\theta(y_i \mid y_{<i}, Q, T; \theta) \quad (1)$$

where θ denotes the parameters of a neural text generation model, and y_i denotes the i-th tokens in the generated answer.

Note that not all semantic units s_i and cell values $c_{i,j}$ contribute to deriving the answer Y. In this paper, we aim to explicitly identify and filter out such irrelevant data prior to reasoning, thereby producing a focused subset $E = \{s_1, s_3, \ldots, s_n\} \subseteq Q$ and a subtable $T_{\text{sub}} \subseteq T$, thereby enabling more efficient and accurate answer derivation.

$$Y = \arg\max \prod_{i=1}^{n} P_\theta(y_i \mid y_{<i}, E, T_{\text{sub}}; \theta) \quad (2)$$

2.2 Model Overview

We propose EnoTab, a dual denoising framework designed to enhance noise robustness in TableQA tasks involving complex questions and large-scale tables. As shown in Figure 2, EnoTab is composed of the Evidence-based Question Denoising (EQD) and the Evidence Tree-guided Table Denoising (ETD). EQD is responsible for removing irrelevant semantic units from the question, while ETD focuses on eliminating irrelevant cell values from the table. More specifically, EQD decomposes the question into a set of evidences, each representing a minimal semantic unit. It then assesses their relevance to answer reasoning based on two criteria, producing a relevant evidence set E_r. Based on this set, ETD builds an explicit table pruning path, called the Evidence Tree. Each step is observable, allowing real-time monitoring and enabling a post-order rollback mechanism to prevent the loss of answer-related data, ultimately yielding a reliable subtable T_{sub}. Finally, EnoTab generates the answer Y to question Q based on the evidence set E_r and the subtable T_{sub}. We next provide a detailed introduction to EQD (Sec. 2.3) and ETD (Sec. 2.4), with further implementation details and hyperparameters available in Appendix A.

2.3 Evidence-based Question Denoising

Complex questions typically contain more semantic units, and both relevant and irrelevant units often appear semantically related to the question, making them hard to distinguish. Therefore, directly identifying irrelevant units based on the entire question is often inaccurate (Ye et al., 2023). To address this, we design *Evidence*, defined as follows.

Definition 1 ***Evidence*** is defined as the representation of a minimal semantic unit in the question that is aligned with a specific data region in the table. Each evidence is formulated as a triplet $e = (\textit{area}, \textit{condition}, \textit{action})$, where *area* specifies the column in the table associated with the semantic unit, *condition* denotes the cell values in that column satisfying the semantic unit, and *action* indicates how the condition should be applied (e.g., string matching, numerical comparison, or date evaluation). For example, for the unit "cities in Tel Aviv", the corresponding evidence links to the `District` column, sets the condition to `Tel Aviv`, and uses `string matching` as the action.

Evidence Generation Unlike subquestions or program code that often involve multiple semantic units (Ye et al., 2023; Nahid and Rafiei, 2024b), each evidence corresponds to only one unit, which

means it is simpler in structure and easier to generate. Specifically, for a complex question Q, we leverage a powerful LLM M_e to generate an evidence set E by decomposing Q, formulated as:

$$E = \{e_1, e_2, \ldots, e_n\} \leftarrow M_e(Q, H, R) \quad (3)$$

where H represents the table header, R denotes k representative rows sampled from the table T. These rows serve as context to help M_e understand the table schema and semantics during evidence generation. Inspired by Chase-SQL (Pourreza et al., 2024), we adopt a two-stage retrieval process. First, an LLM extracts keywords from the question to perform coarse filtering via Locality-Sensitive Hashing (LSH). The retrieved candidates are then re-ranked using embedding similarity and edit distance to select the top-k most relevant rows.

Note that traditional methods focus on using question decomposition to plan reasoning paths (Ye et al., 2023; Wang et al., 2024), while we aim to extract individual semantic units from the question with decomposition as a principled technique, in order to facilitate subsequent relevance assessment.

Evidence Evaluation Given the evidence set E, a natural problem arises: how to assess whether each evidence is relevant to answer reasoning. However, unlike simple questions with ground truth, evidence requires more reliable assessment criteria, which means that approaches such as only using the LLM as a judge are no longer suitable (Lin et al., 2025). Therefore, we design a more comprehensive assessment criterion based on two aspects: consistency and usability.

Specifically, inspired by self-consistency (Wang et al., 2022b), we assume that evidence should appear consistent during the multi-round generation if it is truly relevant to answer reasoning. So, how can we determine whether there is consistent evidence across multi-round generations? Each evidence is defined as a triple $(\textit{area}, \textit{condition}, \textit{action})$ (see Definition 1), where *area* and *action* can be directly compared via string match. However, the same *condition* may appear in different surface forms. For example, in Figure 2, "in Center" and "== Center" are considered consistent. To address this, we use a semantic discriminator M_d such as LLaMA (Grattafiori et al., 2024) to determine whether two conditions are semantically equivalent:

$$\textit{same} \leftarrow M_d(\textit{condition}_1, \textit{condition}_2) \quad (4)$$

where $\textit{same} \in \{\text{True}, \text{False}\}$. Then we generate n rounds of evidence sets and design an efficient algorithm (see Algorithm 1 in the Appendix A.2) to generate a candidate evidence set and compute the consistency score S for each candidate. If $S \geq \alpha$, where α is a predefined threshold, the evidence is retained; otherwise, it is discarded.

How do we define the usability of evidence? As stated in Definition 1, each evidence is expected to be grounded in the table. We consider an evidence instance unusable for answer reasoning if it cannot be matched with any corresponding data in the table, regardless of whether the failure arises from semantic irrelevance or system limitations. To support this process, we design a toolkit $\mathcal{P}$ that integrates multiple APIs to determine whether a given piece of evidence can be grounded in the table. Given an evidence e and a table T as input, $\mathcal{P}$ searches for data in T that satisfies e. If a match is found, it returns `true`; otherwise, it returns `false`. Only evidences for which $\mathcal{P}$ returns `true` are retained for subsequent reasoning. Based on the two criteria described above, we obtain a reliable evidence set E_r.

2.4 Evidence Tree-guided Table Denoising

To further improve the efficiency and effectiveness of reasoning, it is also necessary to remove data in the table that is irrelevant to answer reasoning. Most existing table pruning methods operate in a black-box manner, which increases the risk of losing target data. Therefore, we design the *Evidence Tree*, which constructs an explicit and transparent pruning path based on the reliable evidence set E_r as the filtering criterion. The *Evidence Tree* is defined as follows:

Definition 2 ***Evidence Tree*** is a binary tree (as illustrated in Step 3 of Figure 2), denoted as $\mathcal{T} = (N_{\text{leaf}}, N_{\text{inter}})$. N_{leaf} is the set of leaf nodes, where each node satisfies $n_{\text{leaf}} \in E_r$, meaning that each leaf corresponds to one element in the evidence set. Each leaf node takes the original table as input, filters it based on the corresponding evidence, and outputs a sub-table. N_{inter} is the set of internal nodes, where each node satisfies $n_{\text{inter}} \in \{\text{And}, \text{Or}\}$, indicating the logical relation between its two child nodes. Each internal node takes the two sub-tables from its child nodes as input and outputs a merged sub-table according to the specified logical relation.

Tree Construction We also employ a powerful LLM M_r to generate the evidence tree. Given the table header H, the same k representative rows R, the question Q, and a reliable evidence set E_r, we prompt M_r to produce the evidence tree $\mathcal{T}$. Formally,

$$\mathcal{T} = (N_{\text{leaf}}, N_{\text{inter}}) \leftarrow M_r(Q, H, R, E) \quad (5)$$

Further implementation details are provided in the Appendix A. Since the evidence tree $\mathcal{T}$ is a binary tree, we can adopt post-order traversal to linearize the execution of this inherently non-linear structure. To execute this process, we employ the previously introduced toolkit $\mathcal{P}$. For a leaf node n_{leaf}, $\mathcal{P}$ takes the original table T and the associated evidence e_r as input, and outputs a corresponding sub-table. For an internal node n_{inter}, $\mathcal{P}$ takes the two sub-tables generated by its left and right child nodes, along with the logical operator contained in n_{inter}, and returns a merged table based on that operator. Finally, our goal is to obtain a sub-table T_{sub} by feeding the original table T and the evidence tree $\mathcal{T}$ into the toolkit $\mathcal{P}$.

Subtable Generation During the post-order traversal of the evidence tree $\mathcal{T}$, the resulting sub-table T_{sub} may sometimes be empty. To address this issue, we first analyze the source of the problem. It typically arises at internal nodes n_{inter} with the `AND` operator, since each leaf node n_{leaf} is guaranteed to produce non-empty tables through evidence assessment, and internal nodes n_{inter} with the `OR` operator do not eliminate all entries. We then further examine its underlying causes: (1) critical answer-related data has been lost prior to reaching the current node n_{inter}; or (2) the `AND` operation at the current node n_{inter} yields an empty intersection. To mitigate this, we propose a simple yet effective method called the *And2Or* operation, which replaces `AND` with `OR`. Specifically, we apply the *And2Or* operation sequentially to the left child, right child, and finally the current node. The procedure (illustrated in Figure 2) proceeds as follows: (i) check whether the current node is an internal node with `AND` logic; (ii) if yes, replace `AND` with `OR`; (iii) if a non-empty table is produced, stop; if not, move to the next node. Finally, we ensure that a non-empty subtable T_{sub} is obtained.

To further guarantee the completeness of the target data, we employ a table verifier M_i to determine whether T_{sub} contains all the information required to answer Q. Given T_{sub} as input, M_i returns `True` or `False`. If the result is `True`, T_{sub} is accepted as the final subtable. Otherwise, the process rolls back to the subtable generated at the previous node and re-initiates verification. For cost efficiency, we allow at most two verification attempts. If the second attempt still returns `False`, the full table is used instead. Notably, traditional methods typically regenerate the subtable from scratch when encountering incomplete information (Yu et al., 2025), while our approach enhances efficiency by reusing the previous subtable rather than restarting the process. At the end, we obtain the final subtable T_{final}.

Answer Generation We have removed irrelevant content in the table T and the question Q. In this stage, we perform end-to-end TableQA. Given a subtable T_{final} and a highlighted question Q_{final} using the relevant evidence set E_r, we prompt the LLM to generate the final answer Y.

3 Experiments

3.1 Experimental Setup

Datasets We evaluate EnoTab on four datasets: WikiTQ (Pasupat and Liang, 2015), a table reasoning dataset with 4,344 samples from 421 tables, and TabFact (Chen et al., 2020), a table-based fact verification dataset containing 2,024 samples from 298 tables. In addition, we construct two large-scale table datasets based on Spider (Yu et al., 2018), denoted as STQA-L and STQA-N, corresponding to naturally large tables and noise-injected large tables (Wang et al., 2025a) (construction details are provided in the Appendix A.6).

Baselines We compare our approach against three categories of baselines: (1) **Generic methods** End-to-End QA, Chain-of-Thought, and Text-to-SQL (Rajkumar et al., 2022). (2) **Decomposition-based methods** Dater (Ye et al., 2023), Chain-of-Table (Wang et al., 2024), and TabLaP (Wang et al., 2025b). (3) **Pruning-based methods** Binder (Cheng et al., 2023), TabSQLify (Nahid and Rafiei, 2024b), and H-Star (Abhyankar et al., 2025).

Evaluation Metric For the STQA-N, STQA-L, and WikiTQ datasets, we use exact match accuracy to check whether the predicted answer matches the ground truth. For TabFact, we adopt binary classification accuracy as evaluation metric.

3.2 Main Results

Large-scale Tables Table 1 presents the performance of EnoTab on two large-scale table

Method	STQA-N		STQA-L		Average
	GPT-4o	GPT-4o-mini	GPT-4o	GPT-4o-mini	
End-to-End QA	62.8	57.6	66.4	59.8	61.7
Chain-of-Thought	63.7	59.8	66.7	60.5	62.7
Text-to-SQL (Rajkumar et al., 2022)	61.7	62.6	46.3	47.1	54.4
Binder (Cheng et al., 2023)	64.1	63.9	48.7	45.6	55.6
Dater (Ye et al., 2023)	59.3	54.4	50.4	42.6	51.7
Chain-of-Table (Wang et al., 2024)	59.7	56.9	57.1	57.3	57.8
TabSQLify (Nahid and Rafiei, 2024b)	65.9	61.8	60.6	58.7	61.8
H-Star (Abhyankar et al., 2025)	70.2	68.6	66.2	63.8	67.2
TabLaP (Wang et al., 2025b)	<u>73.6</u>	<u>70.8</u>	<u>69.1</u>	<u>65.9</u>	<u>69.9</u>
EnoTab(Ours)	**80.3** (+8.3)	**78.2** (+9.5)	**75.3** (+8.2)	**72.5** (+9.1)	**76.6** (+8.7)

Table 1: Overall results on two large-scale TableQA datasets STQA-N and STQA-L. The best result is **bold**, and the second-best result is <u>underlined</u>. Numbers in () indicate the performance gain over the second-best method.

datasets: STQA-N and STQA-L. The results show that EnoTab significantly outperforms all baseline methods, demonstrating its effectiveness in large-scale TableQA scenarios. On STQA-N, both decomposition-based and pruning-based methods achieve relatively good performance, with pruning-based methods performing slightly better. We attribute this to the high volume of noisy tokens in STQA-N, which degrades reasoning performance. In contrast, pruning-based methods help alleviate reasoning pressure by removing irrelevant data. Among them, EnoTab stands out by leveraging its powerful and stable pruning capability, achieving state-of-the-art performance.

Furthermore, EnoTab exhibits even greater advantages on the STQA-L dataset, further demonstrating its superiority on truly large-scale tables. The results show that EnoTab achieves more significant performance gains, while pruning-based methods suffer a notable performance drop. We attribute this to the fact that STQA-L consists of naturally large tables with more complex structures and data formats compared to STQA-N, thus requiring more precise pruning. Poor pruning often leads to the loss of answer-related information, resulting in reasoning failures. In contrast, question decomposition methods maintain relatively stable performance, benefiting from the strong reasoning capabilities of the underlying language models. These findings further validate EnoTab's exceptional capability in handling large-scale tables.

Complex Questions To evaluate EnoTab's ability to handle complex questions, we categorize questions in the WikiTQ dataset into difficulty levels based on the reasoning performance of GPT-4o. Each question is independently answered 100 times by GPT-4o and labeled as "Easy" (90–100 correct), "Medium" (60–89), "Hard" (10–59), or "Extra Hard" (0–9). The distribution across difficulty levels is shown in Figure 3. We further analyze the number of evidences generated by EnoTab for questions of different difficulty levels, along with the average per level. Results reveal a clear positive correlation: the more complex the question, the more evidences EnoTab generates, indicating that the method adaptively adjusts its reasoning depth according to question difficulty.

Figure 4 shows the performance comparison between EnoTab and typical decomposition-based methods such as Chain-of-Table and Dater under different difficulty levels using GPT-4o-mini. EnoTab consistently achieves higher accuracy across all levels. In particular, under the "Extra Hard" setting, where baseline performance drops significantly, EnoTab maintains a clear advantage. These results further validate the robustness and reasoning ability of EnoTab in handling complex questions.

Standard Benchmark We further evaluate EnoTab on two standard benchmark datasets, WikiTQ and TabFact, using GPT-4o-mini as the evaluation model. Table 2 shows that EnoTab consistently outperforms existing methods on both datasets. This demonstrates that EnoTab not only excels in substantial noisy data scenarios such as complex questions and large-scale tables, but also maintains strong performance on standard TableQA tasks, indicating good generalizability.

3.3 Experimental Analysis

Ablation Study To quantify the contribution of each component in EnoTab, we conduct ablation studies, as shown in Table 3. Specifically,

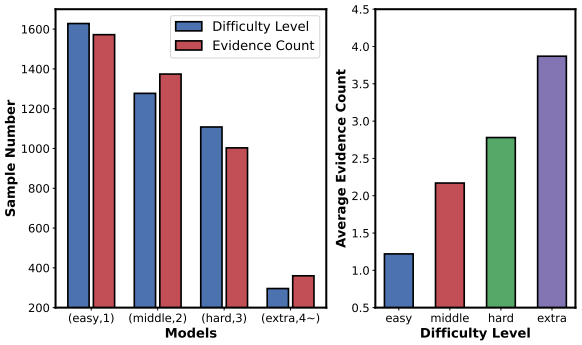

Figure 3: Distribution of question difficulty levels in the WikiTQ dataset and the average number of evidences generated by EnoTab per level.

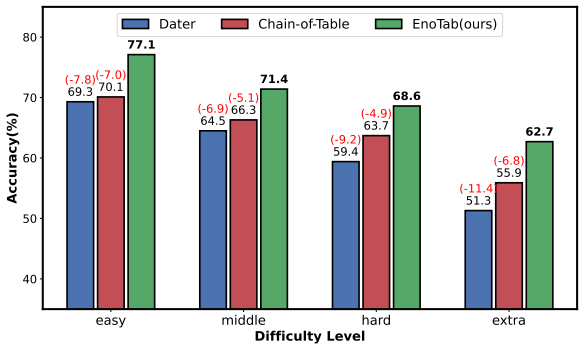

Figure 4: Accuracy comparison of EnoTab, Chain-of-Table, and Dater across different difficulty levels in the WikiTQ dataset, evaluated with GPT-4o-mini.

Method	WikiTQ	TabFact	Average
End-to-End QA	52.6	73.5	63.1
Chain-of-Thought	58.2	77.2	67.7
Text-to-SQL	52.9	69.7	61.3
Binder	58.8	77.2	68.0
Dater	58.3	80.1	69.2
Chain-of-Table	67.1	84.2	75.7
TabSQLify	66.4	78.8	72.6
H-Star	73.8	88.4	81.1
TabLaP	72.8	86.9	79.9
EnoTab(Ours)	**74.6** (+1.8)	**89.2** (+2.3)	**81.9** (+2.0)

Table 2: Performance comparison between EnoTab and previous work on WikiTQ and TabFact datasets. The best result is **bold**, and the second-best result is underlined.

Method	STQA-N	▽	STQA-L	▽
EnoTab(GPT-4o)	**80.3**	–	**75.3**	–
w/o Consistency Assessment	74.0	(-6.3)	69.7	(-5.6)
w/o Usability Assessment	72.1	(-8.2)	68.5	(-6.8)
w/o And2Or Operation	76.1	(-4.2)	71.5	(-3.8)
w/o Table Verifier	75.6	(-4.2)	72.8	(-2.5)
EnoTab(GPT-4o-mini)	**78.2**	–	**72.5**	–
w/o Consistency Assessment	72.7	(-5.5)	66.1	(-6.4)
w/o Usability Assessment	70.3	(-6.9)	65.6	(-6.9)
w/o And2Or Operation	73.8	(-4.4)	68.1	(-4.4)
w/o Table Verifier	74.3	(-3.9)	68.5	(-4.0)

Table 3: Ablation study. Evaluations are conducted on STQA-N and STQA-L using GPT-4o and GPT-4o-mini.

we systematically disable the following key modules: *Consistency Assessment* and *Usability Assessment* from the Evidence Assessment module, and *And2Or Operation* and *Table Verifier* from the Subtable Generation module. Evaluations are performed using GPT-4o-mini on the STQA-L and STQA-N datasets. The results indicate that disabling any single component leads to performance degradation on both datasets, confirming the significant positive impact of each module on the overall effectiveness of the framework. In particular, removing Consistency Assessment or Usability Assessment results in the most noticeable drop, highlighting their critical role in enhancing the framework's relevance filtering capability by accurately identifying and eliminating spurious correlations in the question. Likewise, removing the And2Or Operation or the Table Verifier also causes a substantial decline in performance, demonstrating their importance in improving the framework's table pruning capability and in preventing the inadvertent removal of crucial information during pruning. In summary, the results show that all components in EnoTab play indispensable roles in maintaining system performance. The absence of any module weakens the system's reasoning ability, underscoring the synergy among components and the necessity of their integration for achieving robust performance.

Adaptability To evaluate the adaptability of EnoTab across different types of foundation models, we conduct experiments using both closed-source models (GPT-4o-mini and GPT-4o) and open-source models (LLaMA-2-70B and Qwen-1.5-70B). We randomly sample 200 examples from the WikiTQ dataset and compare the performance of EnoTab with a standard End-to-End QA method. As shown in Figure 5, End-to-End QA suffers a noticeable performance drop when switching from closed-source to open-source models, indicating a strong reliance on the underlying model's reasoning capability. In contrast, EnoTab maintains stable performance across different model configurations, with only minimal degradation even when paired with weaker models. This robustness can be attributed to EnoTab's effective relevance filtering and reliable table pruning capabilities, which jointly elimi-

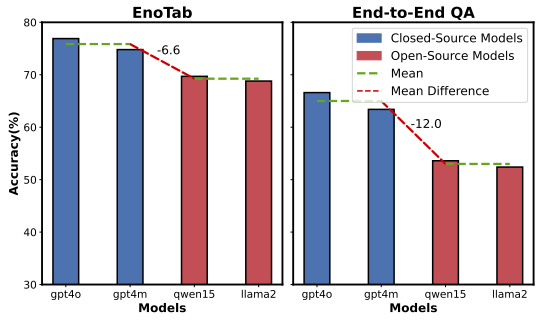

Figure 5: Execution accuracy of closed-source and open-source models on WikiTQ.

Dataset	# Tokens per Table		Comp. (%)
	Entire Table	Pruned Table	
TabFact	343	216	37.0%
WikiTQ	627	294	53.1%
STQA-L	8,967	2,176	75.7%
STQA-N	26,742	2,893	89.2%

Table 4: Token counts and compression rates before and after pruning across four datasets.

nate irrelevant content from both the table and the question. By reducing reasoning burden and complexity, EnoTab enables the model to perform well even under limited capacity. These findings suggest that EnoTab generalizes better across foundation models of varying quality and is more suitable for real-world scenarios where high-performance closed-source models may not always be available.

Table Pruning Effectiveness To evaluate the effectiveness of EnoTab's table pruning capability, we randomly sample 200 correctly answered instances from each of the four datasets and compute the average number of tokens before and after pruning. Table 4 reports the average token counts and corresponding compression rates. For the large-scale datasets STQA-L and STQA-N, EnoTab achieves substantial compression, demonstrating strong pruning ability by effectively removing irrelevant content. Meanwhile, on the relatively smaller-scale datasets WikiTQ and TabFact, EnoTab maintains reliable performance, avoiding the erroneous removal of answer-related data despite the limited presence of noise. These results confirm that EnoTab consistently adapts to varying table scales and exhibits robust and effective table pruning across different scenarios.

4 Related Work

The strong reasoning capabilities of LLMs have significantly advanced the development of TableQA. To further improve table reasoning performance, existing research has primarily explored two key directions (due to space limitations, more related work is discussed in the Appendix D).

Question Decomposition This line of research typically decomposes complex questions into simpler intermediate steps to extend the reasoning chain, thereby easing the burden on LLMs and improving accuracy (Ye et al., 2023; Wang et al., 2024; Zhao et al., 2024; Wu and Feng, 2024). However, these methods heavily rely on the correctness of intermediate steps. As question complexity increases, spurious correlations become more prevalent, causing the reasoning process to deviate from the original intent and potentially leading to failure. In contrast, EnoTab proactively filters spurious correlations before reasoning begins, effectively mitigating such issues and improving overall robustness. Notably, while prior work uses decomposition to construct reasoning paths, EnoTab leverages it as a supporting technique to enable fine-grained assessment of semantic units in complex questions.

Table Pruning Another line of research focuses on pruning tables by generating SQL or Python programs to filter out irrelevant data, producing smaller subtables that reduce reasoning complexity (Zhang et al., 2023; Nahid and Rafiei, 2024b; Abhyankar et al., 2025; Nahid and Rafiei, 2024a; Mao et al., 2024). However, these methods often operate as black boxes, lacking transparency and making it difficult to detect and correct errors in time, which can result in the loss of critical answer-related data. In contrast, EnoTab ensures that each pruning step is observable and verifiable, allowing timely detection and correction of erroneous states.

5 Conclusion

In this paper, we identify the bottlenecks of TableQA in handling complex questions and noisy large-scale tables, which stem from limited relevance filtering and pruning. To tackle these challenges, we present EnoTab, a dual denoising framework that decomposes questions into semantic units, grounds them in tables, and builds explicit reasoning paths with rollback to preserve answer-related information. Extensive experiments across multiple benchmarks demonstrate that EnoTab achieves substantial gains on complex and

large-scale TableQA tasks, highlighting the effectiveness and generality of our approach.

Limitations

EnoTab is currently evaluated in the context of single-table question answering. Although the method is capable of handling multiple tables, its performance in multi-table settings remains unclear. Future work will explore and assess its effectiveness in such scenarios. Additionally, the validation method struggles with certain specialized table formats, such as "*1-1*", where the numbers represent the count of wins and losses. We aim to address this limitation in future work.

Ethics Statement

All datasets used in this study are publicly available through peer-reviewed publications cited in the references. Our framework incorporates GPT-4o and GPT-4o-mini, which may inherit ethical concerns associated with large language models, such as the potential to generate inaccurate or harmful content. We encourage users to critically evaluate the outputs produced by EnoTab before downstream use. Additionally, our method leverages open-source models such as Qwen-1.5 and Llama-2. We adhere to their usage policies and license agreements, and we acknowledge their significant contribution to this research.

Acknowledgement

This work is supported by the National Key R&D Program of China, No. 2024YDLN0004 and the National Natural Science Foundation of China (U23A20275).

References

Nikhil Abhyankar, Vivek Gupta, Dan Roth, and Chandan K Reddy. 2025. H-star: Llm-driven hybrid sql-text adaptive reasoning on tables. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8841–8863.

Shengnan An, Xunliang Cai, Xuezhi Cao, Xiaoyu Li, Yehao Lin, Junlin Liu, Xinxuan Lv, Dan Ma, Xuanlin Wang, Ziwen Wang, et al. 2025. Amo-bench: Large language models still struggle in high school math competitions. *arXiv preprint arXiv:2510.26768*.

Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 1(1):538–549.

Si-An Chen, Lesly Miculicich, Julian 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. *Advances in Neural Information Processing Systems*, 37:74899–74921.

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, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Binding language models in symbolic languages. *ICLR*, abs/2210.02875.

Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2022. Turl: Table understanding through representation learning. *ACM SIGMOD Record*, 51(1):33–40.

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

Zihui Gu, Ju Fan, Nan Tang, Preslav Nakov, Xiaoman Zhao, and Xiaoyong Du. 2022. Pasta: table-operations aware fact verification via sentence-table cloze pre-training. *arXiv preprint arXiv:2211.02816*.

Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349*.

Rohit Khoja, Devanshu Gupta, Yanjie Fu, Dan Roth, and Vivek Gupta. 2025. Weaver: Interweaving sql and llm for table reasoning. *arXiv preprint arXiv:2505.18961*.

Mike Lewis. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Honglin Lin, Zheng Liu, Yun Zhu, Chonghan Qin, Juekai Lin, Xiaoran Shang, Conghui He, Wentao Zhang, and Lijun Wu. 2026a. Mmfinereason: Closing the multimodal reasoning gap via open data-centric methods. *arXiv preprint arXiv:2601.21821*.

Honglin Lin, Chonghan Qin, Zheng Liu, Qizhi Pei, Yu Li, Zhanping Zhong, Xin Gao, Yanfeng Wang, Conghui He, and Lijun Wu. 2026b. Scientific image synthesis: Benchmarking, methodologies, and downstream utility. *arXiv preprint arXiv:2601.17027*.

Xin Lin, Zhenya Huang, Zhiqiang Zhang, Jun Zhou, and Enhong Chen. 2025. Explore what llm does not know in complex question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24585–24594.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2021. Tapex: Table pre-training via learning a neural sql executor. *arXiv preprint arXiv:2107.07653*.

Zheng Liu, Hao Liang, Xijie Huang, Wentao Xiong, Qinhan Yu, Linzhuang Sun, Chong Chen, Conghui He, Bin Cui, and Wentao Zhang. 2024. Synthvlm: High-efficiency and high-quality synthetic data for vision language models. *arXiv preprint arXiv:2407.20756*, 3.

Zheng Liu, Honglin Lin, Chonghan Qin, Xiaoyang Wang, Xin Gao, Yu Li, Mengzhang Cai, Yun Zhu, Zhanping Zhong, Qizhi Pei, et al. 2026. Chartverse: Scaling chart reasoning via reliable programmatic synthesis from scratch. *arXiv preprint arXiv:2601.13606*.

Zheng Liu, Mengjie Liu, Jingzhou Chen, Jingwei Xu, Bin Cui, Conghui He, and Wentao Zhang. 2025a. Fusion: Fully integration of vision-language representations for deep cross-modal understanding. *arXiv preprint arXiv:2504.09925*.

Zheng Liu, Mengjie Liu, Siwei Wen, Mengzhang Cai, Bin Cui, Conghui He, and Wentao Zhang. 2025b. From uniform to heterogeneous: Tailoring policy optimization to every token's nature. *arXiv preprint arXiv:2509.16591*.

Qingyang Mao, Qi Liu, Zhi Li, Mingyue Cheng, Zheng Zhang, and Rui Li. 2024. Potable: Programming standardly on table-based reasoning like a human analyst. *arXiv preprint arXiv:2412.04272*.

Md Nahid and Davood Rafiei. 2024a. Normtab: Improving symbolic reasoning in llms through tabular data normalization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3569–3585.

Md Nahid and Davood Rafiei. 2024b. Tabsqlify: Enhancing reasoning capabilities of llms through table decomposition. 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 5725–5737.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, et al. 2022. Fetaqa: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2024. Gpt-4 technical report.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.

Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeounoh Chung, Shayan Talaei, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan O Arik. 2024. Chase-sql: Multi-path reasoning and preference optimized candidate selection in text-to-sql. *arXiv preprint arXiv:2410.01943*.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498*.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2024. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 645–654.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Lihan Wang, Bowen Qin, Binyuan Hui, Bowen Li, Min Yang, Bailin Wang, Binhua Li, Jian Sun, Fei Huang, Luo Si, et al. 2022a. Proton: Probing schema linking information from pre-trained language models for text-to-sql parsing. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1889–1898.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022b. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Yuxiang Wang, Junhao Gan, and Jianzhong Qi. 2025a. Tabsd: Large free-form table question answering with sql-based table decomposition. *arXiv preprint arXiv:2502.13422*.

Yuxiang Wang, Jianzhong Qi, and Junhao Gan. 2025b. Accurate and regret-aware numerical problem solver for tabular question answering. In *Proceedings of*

the AAAI Conference on Artificial Intelligence, volume 39, pages 12775–12783.

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. 2024. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *ICLR*.

Fei Wu, Zhenrong Zhang, Qikai Chang, Jianshu Zhang, Quan Liu, and Jun Du. 2026. Step potential advantage estimation: Harnessing intermediate confidence and correctness for efficient mathematical reasoning. *arXiv preprint arXiv:2601.03823*.

Zhenhe Wu, Zhongqiu Li, Mengxiang Li, Jie Zhang, Zhongjiang He, Jian Yang, Yu Zhao, Ruiyu Fang, Yongxiang Li, Zhoujun Li, et al. 2025a. Mr-sql: multi-level retrieval enhances inference for llm in text-to-sql. In *International Conference on Database Systems for Advanced Applications*, pages 403–413. Springer.

Zhenhe Wu, Zhongqiu Li, Jie Zhang, Zhongjiang He, Jian Yang, Yu Zhao, Ruiyu Fang, Bing Wang, Hongyan Xie, Shuangyong Song, and Zhoujun Li. 2025b. UCS-SQL: Uniting content and structure for enhanced semantic bridging in text-to-SQL. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8156–8168, Vienna, Austria. Association for Computational Linguistics.

Zhenhe Wu, Jian Yang, Jiaheng Liu, Xianjie Wu, Changzai Pan, Jie Zhang, Yu Zhao, Shuangyong Song, Yongxiang Li, and Zhoujun Li. 2025c. Table-r1: Region-based reinforcement learning for table understanding. *arXiv preprint arXiv:2505.12415*.

Zirui Wu and Yansong Feng. 2024. Protrix: Building models for planning and reasoning over tables with sentence context. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4378–4406.

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*, pages 174–184.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314*.

Peiying Yu, Guoxin Chen, and Jingjing Wang. 2025. Table-critic: A multi-agent framework for collaborative criticism and refinement in table reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17432–17451, Vienna, Austria. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Wenyuan Zhang, Xinghua Zhang, Haiyang Yu, Shuaiyi Nie, Bingli Wu, Juwei Yue, Tingwen Liu, and Yongbin Li. 2026. Expseek: Self-triggered experience seeking for web agents.

Xuanliang Zhang, Dingzirui Wang, Longxu Dou, Qingfu Zhu, and Wanxiang Che. 2025. A survey of table reasoning with large language models. *Frontiers of Computer Science*, 19(9):199348.

Yunjia Zhang, Jordan Henkel, Avrilia Floratou, Joyce Cahoon, Shaleen Deep, and Jignesh M Patel. 2023. Reactable: Enhancing react for table question answering. *arXiv preprint arXiv:2310.00815*.

Yilun Zhao, Lyuhao Chen, Arman Cohan, and Chen Zhao. 2024. Tapera: enhancing faithfulness and interpretability in long-form table qa by content planning and execution-based reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12824–12840.

Wei Zhou, Mohsen Mesgar, Annemarie Friedrich, and Heike Adel. 2025a. Efficient multi-agent collaboration with tool use for online planning in complex table question answering. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 945–968.

Yixiao Zhou, Yang Li, Dongzhou Cheng, Hehe Fan, and Yu Cheng. 2026. Look inward to explore outward: Learning temperature policy from llm internal states via hierarchical rl. *arXiv preprint arXiv:2602.13035*.

Yixiao Zhou, Ziyu Zhao, Dongzhou Cheng, Zhiliang Wu, Jie Gui, Yi Yang, Fei Wu, Yu Cheng, and Hehe Fan. 2025b. Dropping experts, recombining neurons: Retraining-free pruning for sparse mixture-of-experts llms. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 15169–15186.

A Additional Implementation Details of EnoTab

This section provides additional implementation details of EnoTab to supplement the high-level overview in the main text. We elaborate on the design of key components, including evidence generation, evidence evaluation, Evidence Tree construction, and rollback, along with the associated prompting strategies and parameter configurations. These details aim to clarify the complete pipeline and support both transparency and reproducibility.

A.1 Two-Stage Retrieval Process

During the evidence generation phase, directly using the entire table to produce evidence is computationally inefficient and prone to errors, particularly when the table contains thousands of rows. Selecting a small set of representative rows that are most relevant to the question is typically sufficient for generating high-quality evidence. To achieve this, we adopt a two-stage retrieval strategy that balances efficiency and accuracy. The process consists of keyword-based filtering followed by embedding-based re-ranking.

Stage 1: Keyword-based Filtering. Given a question Q, we first extract a set of task-relevant keywords $\{k_1, k_2, \ldots, k_m\}$ using a lightweight language model. We then apply LSH to retrieve candidate rows R_c from the table T that exhibit substantial token overlap with the extracted keywords. This step significantly reduces the candidate space from thousands of rows to a much smaller subset.

Stage 2: Embedding-based Re-ranking. For each candidate row $r \in R_c$, we compute its semantic similarity with the question Q using a pre-trained embedding model $f(\cdot)$. To enhance robustness, we also incorporate a lexical similarity score based on the edit distance between the row tokens and the extracted keywords. The final ranking score for row r is defined as:

$$\text{Score}(r, Q) = \lambda \cdot S_{\text{sem}}(r, Q) + (1 - \lambda) \cdot S_{\text{lex}}(r, Q),$$

where S_{sem} denotes semantic similarity measured via cosine similarity of embeddings, and S_{lex} denotes lexical similarity based on edit distance.

In practice, we set $k = 10$, $\lambda = 0.7$, and $C = \min(256, \lceil 0.1N \rceil)$, where N is the number of rows in the table. We use GPT-4o-mini for keyword extraction and `bge-large-en-v1.5` as the default embedding encoder. This two-stage retrieval strategy significantly improves the efficiency of evidence generation.

A.2 Consistency Assessment of Evidence

The consistency assessment process of evidence is presented in Algorithm 1. Following (Lin et al., 2025), we set the number of rounds to $n = 5$ and the threshold to $\alpha = 0.8$. For semantic discrimination, we adopt Llama-2-7b-chat-hf as the default discriminator M_d, which strikes a balance between accuracy and efficiency.

Algorithm 1: Evidence Consistency Assessment

Input: n rounds of evidence sets $\mathcal{E} = \{E_1, E_2, ..., E_n\}$, threshold α, semantic discriminator M_d
Output: Candidate evidence set E_c

Initialize empty multimap $\mathcal{G}$ for grouping evidence by (area, action);
$E_c \leftarrow \emptyset$;
foreach *evidence set* $E_i \in \mathcal{E}$ **do**
  foreach *evidence* $e \in E_i$ **do**
    Extract $(\text{area}_e, \text{condition}_e, \text{action}_e)$;
    Append e to group $\mathcal{G}[(\text{area}_e, \text{action}_e)]$;
  end
end
foreach *group* G_j *in* $\mathcal{G}$ **do**
  foreach *evidence* $e_i \in G_j$ **do**
    $c \leftarrow 0$;
    foreach *evidence* $e_k \in G_j$, $e_k \neq e_i$ **do**
      if $M_d(\textit{condition}_{e_i}, \textit{condition}_{e_k}) =$ *True* **then**
        $c \leftarrow c + 1$;
      end
    end
    Compute consistency score $S(e_i) \leftarrow \frac{c}{|G_j|-1}$;
    if $S(e_i) \geq \alpha$ **then**
      Add e_i to E_c;
    end
  end
end
return E_c;

A.3 Usability Assessment of Evidence

In this section, we describe how to verify the *usability* of each evidence using the toolkit P (as shown in Figure 6). Given an evidence $e = (\text{area}, \text{condition}, \text{action})$ and a table T, toolkit P checks whether e can be grounded in T. The process is as follows: (1) select the target column according to `area`; (2) normalize the `condition` into the canonical form required by the selected action; (3) invoke the API corresponding to `action`, such as string matching, numeric comparison, or date evaluation; (4) return whether the resulting sub-table is empty. If the result is non-empty, the

evidence is regarded as *usable*; otherwise, it is discarded.

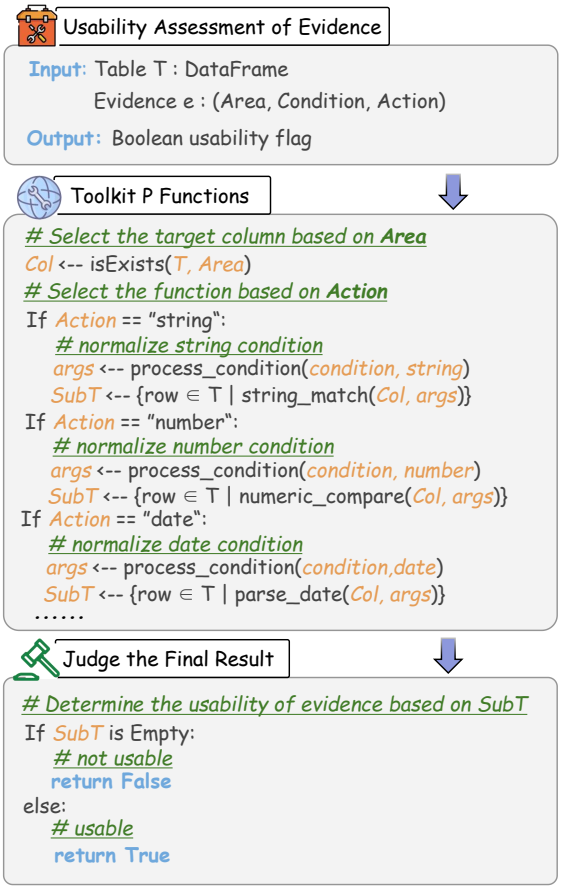

Figure 6: Usability Assessment of Evidence.

A.4 Detailed Description of the Evidence Tree

In this section, we provide a more intuitive explanation of how the Evidence Tree operates. As illustrated in Figure 7, the Evidence Tree is structured as a binary tree consisting of four leaf nodes and three internal nodes. The entire tree mirrors the logical structure implicitly contained in the question and is executed following a post-order traversal strategy.

Figure 8 further illustrates the functional role of leaf nodes and internal nodes. Each leaf node corresponds to a minimal evidence unit and applies a fine-grained filtering operation on the original table. The internal nodes act as logical operators (e.g., AND, OR) that merge the filtered subtables produced by their children. Through this layered design, the pruning process becomes both stepwise and interpretable: every pruning decision is transparent and can be independently inspected.

Importantly, all leaf-level filtering is performed before any internal merging, which avoids prematurely combining incomplete subtables. At each step, the intermediate subtable remains observable, enabling timely detection of abnormal states (e.g., empty tables) and facilitating recovery strategies such as rollback. This property distinguishes the Evidence Tree from black-box pruning approaches and ensures that the pruning trajectory remains auditable and robust.

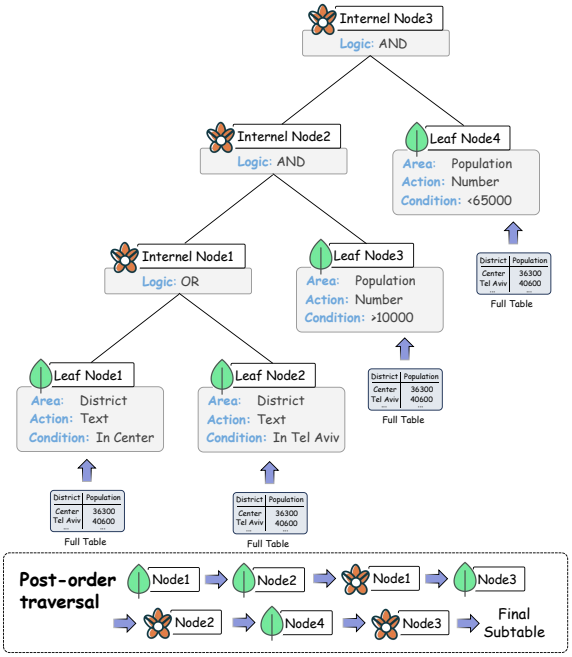

Figure 7: An illustrative Evidence Tree with four leaf nodes and three internal nodes.

A.5 Rollback Mechanism: Analysis of And2Or

The rollback mechanism is designed to prevent the loss of essential information during table pruning. In particular, the And2Or operation acts as a fallback strategy when an internal node with an AND operator produces an empty subtable. This situation indicates that the system has reached the limits of its discriminative capability, being unable to determine which records are relevant. To avoid discarding all potentially useful data, the AND operator is replaced with an OR operator, resulting in a superset that may include redundant or irrelevant rows. While this relaxation may reduce precision, it preserves answer-critical content and ensures that downstream reasoning has sufficient evidence to proceed. In the context of TableQA, recall is generally more important than precision, as irrelevant rows can be filtered later, whereas missing evidence cannot be recovered. In practice, the And2Or opera-

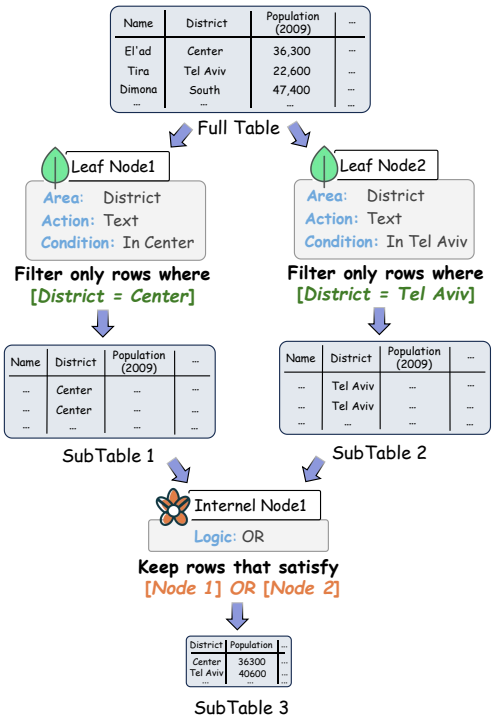

Figure 8: Table transformation process at leaf and internal nodes of the Evidence Tree.

tion helps maintain non-empty intermediate results, allowing later verification modules to assess completeness and trigger corrections when necessary. This mechanism contributes to greater robustness compared to approaches that discard all content under overly strict conditions. In future work, we aim to enhance the model's discriminative ability at conjunction nodes, reducing reliance on And2Or while preserving its effectiveness as a safeguard.

A.6 Details of Dataset Construction

We construct two evaluation sets, STQA-L and STQA-N, from source tables through a semi-automatic QA generation pipeline with manual verification. For each source table, we randomly sample a region, which can be a single cell, a row, or a sub-table consisting of multiple rows and/or columns. The sampled region is not directly used as the final answer. Instead, we prompt an LLM to generate an answer conditioned on the sampled region and then construct a corresponding question. All generated QA pairs are manually reviewed and verified for correctness. Problematic cases identified during review are either corrected or discarded. More than 80% of QA pairs are accepted without edits, while fewer than 20% require manual refinement. We consider three sampling granularities

Dataset	Correct	Answer Error	Invalid Question
STQA-L	81.2%	12.7%	6.1%
STQA-N	80.7%	16.4%	2.9%

Table 5: Statistics of QA construction outcomes.

Dataset	# QA Pairs	# Tokens per Table	# Tokens per Answer
STQA-L	1,074	9,786	3.9
STQA-N	617	28,652	3.1

Table 6: Statistics of QA pairs and token counts per table and answer.

in this process: cell-based sampling selects a single cell, row-based sampling selects an entire row, and subtable-based sampling selects a multi-row, multi-column sub-table.

For STQA-N, we first expand tables shorter than 4,096 tokens by prompting an LLM to generate additional rows and columns, and then construct QA pairs using the same procedure as in STQA-L. After identifying the gold answer cells and columns, we perturb 15% of the non-answer cells with type-specific noise. For string cells, including dates and booleans, we replace the original value with one of the top five semantically closest WordNet alternatives. For numeric cells, we replace the original value with a value outside the gold range while keeping answer correctness unchanged. All QA pairs are then manually verified to ensure that the injected noise does not compromise validity. This process creates realistic distractors and enables evaluation of noise sensitivity.

During QA construction, we observe two major error types. The first is *Answer Error*, where the generated question is correct but the answer is inaccurate; such QA pairs are retained and manually corrected based on the table contents. The second is *Invalid Question*, where the question is ill-posed, ambiguous, or inconsistent with the table; such QA pairs are discarded entirely. Table 5 summarizes the statistics of QA construction outcomes, and Table 6 reports the final dataset statistics.

B Additional Experiments

B.1 Hyper-parameter Settings

Table 7 summarizes the models and parameter settings used in each module of EnoTab. To ensure the stability of our results, we also repeated experiments with different random seeds. For open-source models (e.g., LLaMA-2, Qwen-1.5), we

Description	Value / Model
Top-k representative rows	$k = 10$
Candidate pool size	$C = \min(256, \lceil 0.1N \rceil)$
Ranking weight	$\lambda = 0.7$
Keyword extraction model	GPT-4o-mini
Embedding encoder	`bge-large-en-v1.5`
Consistency rounds	$n = 5$
Consistency threshold	$\alpha = 0.8$
Semantic discriminator	Llama-2-7b-chat-hf

Table 7: Summary of hyper-parameter settings for EnoTab.

Method	FeTaQA			
	BLEU	R-1	R-2	R-L
T5-small	-	55	33	47
T5-base	-	61	39	51
T5-large	-	63	41	53
End-to-End QA	28.37	63	41	53
Dater	29.47	63	41	53
Chain-of-Table	32.61	66	44	56
EnoTab(ours)	**30.46**	**67**	**45**	**57**

Table 8: Performance comparison between EnoTab and previous work on the FeTaQA dataset (evaluated using GPT-3.5-turbo). BLEU and ROUGE scores are reported: R-1, R-2, and R-L denote ROUGE-1, ROUGE-2, and ROUGE-L respectively.

report the average accuracy over 3 runs. For closed-source APIs (e.g., GPT-4o, GPT-4o-mini), which show minimal stochastic variation, we conducted 2 runs to confirm stability. Across all datasets, the results were highly stable, with the standard deviation consistently within 1% absolute accuracy. This demonstrates that the reported improvements of EnoTab are robust and not dependent on random factors.

B.2 Error Analysis

Figure 9 presents our error analysis of the traditional table pruning method Text2SQL on the WikiTQ dataset. The errors are categorized into three types: BinderException, which indicates semantic errors such as referencing non-existent columns, tables, or aliases; ParserException, which refers to SQL syntax errors that prevent parsing; and IndexError, which occurs when execution attempts to access out-of-range rows (e.g., accessing row 10 in a 5-row table). Among these, BinderException is the most frequent. This type of error often arises when any condition in the SQL query is invalid, rendering the entire query unusable and causing pruning to fail (see case in Figure 23). ParserException is the second most common and typically results from conditions or table content that exceed the expressive capacity of SQL (see case in Figure 24).

Our proposed method, EnoTab, effectively addresses these issues. Since each evidence unit is treated as an independent and minimal semantic element, EnoTab decouples multiple SQL conditions into discrete pieces of evidence. Each piece is individually verified and executed, allowing invalid evidence to be filtered out and significantly reducing the occurrence of BinderException. Moreover, EnoTab is able to recognize when certain evidence exceeds its table reasoning capability, thereby avoiding ParserException by design.

While EnoTab is robust in handling large-scale tables, its performance is still limited when dealing with structurally complex tables. For example, some cells contain compound values such as "1–1" (indicating 1 win and 1 loss) or "251–32=189" (where 189 is the value of interest). Successfully pruning such tables requires accurately extracting the relevant data from compound entries. Although equipping the model with this ability could further improve pruning effectiveness, it also introduces a higher risk of losing critical answer-related data. This conflicts with the core principle of EnoTab: preserving target data while pruning as aggressively as possible.

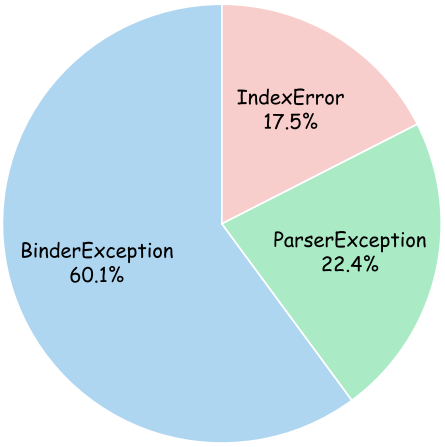

Figure 9: Statistics for the SQL execution errors.

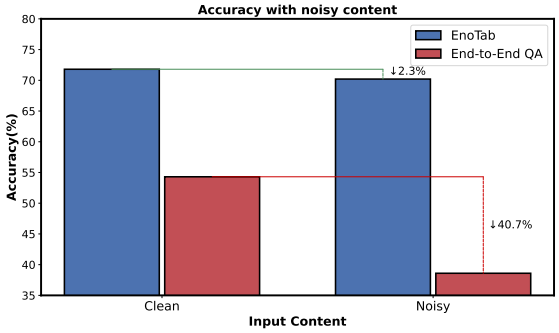

Figure 10: Execution accuracy on WikiTQ with noisy content in tables.

B.3 Noisy Content

To further evaluate EnoTab's robustness to noisy data, we follow the approach proposed in Binder(Cheng et al., 2023) and construct a noisy version of the WikiTQ development set by injecting distractive content to simulate misleading information in real-world scenarios. We evaluate both EnoTab and a standard End-to-End QA method under two settings: "clean" and "noisy". As shown in Figure 10, the performance of End-to-End QA drops significantly when noise is introduced, indicating its sensitivity to misleading content. In contrast, EnoTab maintains stable performance even under noisy conditions, demonstrating stronger robustness. This advantage primarily stems from its efficient table pruning capability, which enables the model to effectively identify and filter out irrelevant information, thereby sustaining high reasoning accuracy even in the presence of noise.

B.4 Experiments of EnoTab on FetaQA

Table 8 shows that EnoTab achieves strong performance on the FeTaQA(Nan et al., 2022) dataset for free-form question answering, consistently outperforming existing state-of-the-art methods. In particular, EnoTab surpasses all baselines on ROUGE-1/2/L(Lin, 2004), which measure lexical overlap (unigram and bigram) and sequence-level similarity based on the longest common subsequence. We attribute this improvement to EnoTab's ability to better capture key information from the table and maintain structural alignment in the generated answers through fine-grained evidence selection and reasoning. We also observe that EnoTab slightly underperforms Chain-of-Table(Wang et al., 2024) in terms of BLEU(Papineni et al., 2002) score. Since BLEU places greater emphasis on n-gram precision and is sensitive to word order, we believe this is due to the fact that our model does not explicitly optimize for surface-level phrasing or sequence alignment. However, manual inspection confirms that the generated answers remain accurate and complete in content, indicating that the performance drop in BLEU does not reflect true semantic degradation.

Dataset	Evidence Count			
	1	2	3	>4
WikiTQ	1628	1277	1108	296
TabFact	1134	573	287	29

Table 9: Number of samples in the WikiTQ and TabFact dataset by evidence count.

Dataset	Difficulty Level			
	easy	middle	hard	extra
WikiTQ	1572	1374	1003	360
TabFact	1347	463	171	42

Table 10: Number of samples in the WikiTQ and TabFact dataset by difficulty level.

B.5 Effectiveness Analysis of Evidence Generation

To evaluate the effectiveness of our evidence generation module, we analyze the relationship between the number of generated evidence pieces and task difficulty across two benchmarks: WikiTQ and TabFact. As shown in Tables 9 and 10, the distribution of samples across different evidence counts closely mirrors the distribution across difficulty levels. This alignment suggests that tasks requiring more evidence are generally more difficult. More notably, we observe that the number of samples with higher evidence counts increases with task difficulty. For instance, in WikiTQ, the proportion of samples requiring 4 or more evidence pieces rises as we move from 'easy' to 'extra' difficulty levels. A similar trend is evident in TabFact, albeit with fewer high-difficulty samples. This increasing evidence requirement reflects the heightened complexity of reasoning in harder tasks and demonstrates that our evidence generation module is sensitive to task difficulty. These findings confirm that our module can effectively capture and extract more granular, informative evidence as the reasoning complexity increases. In other words, it scales appropriately with task demands and is particularly beneficial for handling more challenging questions.

C Prompt

C.1 Prompt of Dataset Construction

The prompt used in data construction, as shown in Figure 11, 12, 13, 14 and 15.

C.2 Prompt of EnoTab

The prompt used in the EnoTab pipeline, as shown in Figure 16, 17, 18 and 19.

C.3 Example of Evidence

The types of Evidence include textual, numerical and date, with corresponding examples shown in Figures 20, 21 and 22, respectively.

D Additional Related Work

LLM Reasoning In recent years, the rapid progress of large language models (LLMs) has made reasoning a core capability for modern AI systems. Reasoning-based paradigms have been widely adopted across diverse applications, including vision-language modeling (Liu et al., 2024, 2025a; Lin et al., 2026a,b), chart understanding (Liu et al., 2026), mathematical problem solving (Wu et al., 2026; An et al., 2025), and web-based autonomous agents (Zhang et al., 2026). Meanwhile, emerging evidence suggests that reasoning processes are highly heterogeneous across different tokens, modules, modalities, and intermediate steps, calling for more adaptive reasoning, optimization, and inference strategies (Liu et al., 2025b; Zhou et al., 2025b, 2026). Against this backdrop, structured reasoning tasks have attracted increasing attention, including table reasoning and closely related text-to-SQL settings, where models must jointly understand structured schemas, content, and compositional reasoning procedures over tabular evidence (Wu et al., 2025b,a,c).

Finetune-based Table Reasoning In the field of table reasoning, early methods typically enhance models' understanding of tables by fine-tuning pre-trained models. The core challenge lies in enabling models to better comprehend the content and structure of tables during training. TaPas(Herzig et al., 2020) improves understanding of tabular data by recovering masked cell information in tables. TABERT(Yin et al., 2020) proposes the concept of content snapshots to encode the most relevant table content subsets based on the input utterance. TURL(Deng et al., 2022) focuses on table relationship understanding by introducing table context information and modeling cell semantics, significantly enhancing table semantic understanding and reasoning. TAPEX(Liu et al., 2021) leverages a BART(Lewis, 2019) model to simulate a SQL executor during pre-training, equipping TAPEX with stronger table reasoning capabilities. PASTA(Gu et al., 2022) introduces an operation-aware fact verification approach, pre-training the language model to learn common table-based operations and solve sentence-table cloze tasks synthesized from WikiTables(Pasupat and Liang, 2015), further improving reasoning capabilities. Additionally, (Sui et al., 2024) addresses large-scale tables by defining a series of constraints to control table size while minimizing the loss of key information.

Pruning and Planning Table Reasoning Some recent studies have attempted to jointly address table pruning and reasoning planning by integrating evidence selection with structured inference control. These approaches aim to reduce reasoning complexity by eliminating irrelevant data while also guiding the model through a well-defined reasoning path. For example, some frameworks alternate between executing partial programs and making intermediate decisions, allowing the reasoning process to adapt dynamically to the evolving context (Khoja et al., 2025). Other methods introduce symbolic planners or controller modules that determine the order in which subtasks—such as filtering, aggregation, or comparison—should be executed (Zhang et al., 2023; Mao et al., 2024). While effective in structured reasoning, such methods often operate as black boxes, making it difficult to trace or correct errors in pruning or planning. In contrast, our method explicitly separates pruning and planning stages, and further introduces verifiable checkpoints at each step to enhance transparency and robustness.

Prompt of Cell-based QA Generation

Instruction:

You are a QA generation expert. Your task is to Randomly select one cell in the table as the answer and generate a question which can produce the answer. Return both the question and answer. Repeat this process for 10 times and return me with 10 QA pairs.

Table:

Header: [HEADER]
Content: [TAB]

Note:

- Directly returns questions in format "Q: question_content; A: answer_content" (without any explanation).
-Try to make the question diverse.
-Keep answers as concise as possible and only contain entities.

Figure 11: Prompt of Cell-based QA Generation.

Prompt of Row-based QA Generation

Instruction:

You are a QA generation expert. Your task is to use selected row above to generate one question using the information within this row. Return the question and its answer. Repeat this process 4 times.

Table:

Header: [HEADER]
Content: [TAB]
Selected Row: [ROW]

Note:

-Directly returns questions in format "Q: question_content; A: answer_content" (without any explanation).
-Try to make the question diverse.
-Do not include "selected row/given data" in questions, as the selected columns are not known to the person answering the question.
-Keep answers as concise as possible and only contain entities.

Figure 12: Prompt of Row-based QA Generation.

Prompt of Column-based QA Generation

Instruction:

You are a QA generation expert. Your task is to use selected column above to generate one question using the information within this column. Return the question and its answer. Repeat this process 4 times.

Table:

Header: [HEADER]
Content: [TAB]
Selected Column: [COL]

Note:

-Directly returns questions in format "Q: question_content; A: answer_content" (without any explanation).
-Try to make the question diverse.
-Do not include "selected column/given data" in questions, as the selected columns are not known to the person answering the question.
-Keep answers as concise as possible and only contain entities.

Figure 13: Prompt of Colmn-based QA Generation.

Prompt of Subtable-based QA Generation

Instruction:

You are a QA generation expert. Your task is to use sub-table above to generate one question using the information within this sub-table. Return the question and its answer. Repeat this process 4 times.

Table:

Header: [HEADER]
Content: [TAB]
Selected Subtable: [SUB]

Note:

-Directly returns questions in format "Q: question_content; A: answer_content" (without any explanation).
-Try to make the question diverse.
-Do not include "sub-table/given data" in questions, as the sub-table is not known to the person answering the question.
-Keep answers as concise as possible and only contain entities.

Figure 14: Prompt of Subtable-based QA Generation.

Prompt of Table expansion with noise

Instruction:
You are a data augmentation expert. Your task is to generate noisy data to expand the provided table both row-wise and column-wise.

Table:
Header: [HEADER]
Content: [TAB]

Note:
- return the entire table directly (no additional explanation).
- make sure there are no duplicate rows or columns.
- expand the table as much as possible to reach approximately 20,000 tokens.

Figure 15: Prompt of Table expansion with noise.

Prompt of Evidence Generation

Instruction:
You are a task requirement understanding expert. Your task is to extract multiple pieces of evidence from the provided table and the question.

Table:
col : Common name | District | Hebrew | Arabic | Population\n(2009) | Area\n(km²) | Mayor
row 1 : Acre | North | עכו | عكا | 46,300 | 13.533 | Shimon Lancry
row 2 : Afula | North | עפולה | العفولة | 40,500 | 26.909 | Avi Elkabetz
row 3 : Arad | South | ערד | عراد | 23,400 | 93.140 | Tali Ploskov
row 4 : Ariel | Judea & Samaria\n(West Bank) | אריאל | أريئل | 17,600 | 14.677 | Eliyahu Shaviro
row 5 : Ashdod | South | אשדוד | أشدود | 206,400 | 47.242 | Yehiel Lasri
row 6 : Ashkelon | South | אשקלון | عسقلان | 111,900 | 47.788 | Benny Vaknin
.....

Question:
How many cities in Tel Aviv or Center have a population between 10,000 and 65,000 and an area of less than 40?

Response:
area[colmn2:District] action[textural] condition[Center in x]
area[colmn2:District] action[textural] condition[Tel Aviv in x]
area[colmn3:Population\n(2009)] action[numerical] condition[x < 65000]
area[colmn3:Population\n(2009)] action[numerical] condition[x > 10000]
area[colmn4:Area] action[numerical] condition[x < 40]

Figure 16: Prompt of Evidence Generation.

Prompt of Tree Construction

Instruction:
You are a task requirement understanding expert. Your task is to link the relationships between the provided pieces of evidence. Then generate an Evidence Graph.

Table:
col : Common name | District | Hebrew | Arabic | Population\n(2009) | Area\n(km²) | Mayor
row 1 : Acre | North | עכו | عكا | 46,300 | 13.533 | Shimon Lancry
row 2 : Afula | North | עפולה | العفولة | 40,500 | 26.909 | Avi Elkabetz
row 3 : Arad | South | ערד | عراد | 23,400 | 93.140 | Tali Ploskov
row 4 : Ariel | Judea & Samaria\n(West Bank) | אריאל | أريال | 17,600 | 14.677 | Eliyahu Shaviro
row 5 : Ashdod | South | אשדוד | أشدود | 206,400 | 47.242 | Yehiel Lasri
row 6 : Ashkelon | South | אשקלון | عسقلان | 111,900 | 47.788 | Benny Vaknin
.....

Question:
How many cities in Tel Aviv or Center have a population between 10,000 and 65,000 and an area of less than 40?

Evidence:
E1:area[colmn2:District] action[textural] condition[Center in x]
E2:area[colmn2:District] action[textural] condition[Tel Aviv in x]
E3:area[colmn3:Population\n(2009)] action[numerical] condition[x < 65000]
E4:area[colmn3:Population\n(2009)] action[numerical] condition[x > 10000]
E5:area[colmn4:Area] action[numerical] condition[x < 40]

Response:
((((E1 OR E2) AND E3) AND E4) AND E5)

Figure 17: Prompt of Tree Construction.

Prompt of Final Query

Instruction:
You are a table question answering expert. Your task is to infer the answer to the question based on the provided table.

Example:
.....
.....
.....

Table:
col : District | Population\n(2009) | Area\n(km²)
row 1 : Center | 36300 | 2.756
row 2 : Tel Aviv | 40600 | 16.792
row 3 : Tel Aviv | 34400 | 5.141
row 4 : Tel Aviv | 31000 | 4.112

Question:
How many cities in Tel Aviv or Center have a population between 10,000 and 65,000 and an area of less than 40?
highlight semantic units:
[in Tel Aviv or Center] [population between 10,000 and 65,000] [area of less than 40]

Response:
4

Figure 18: Prompt of Final Query.

Prompt of Table Verifier

Instruction:
You are a TableQA expert. Your task is to determine whether the following pruned subtable is sufficient to answer the given question.

Input
Question: **[Q]**
Subtable: **[T]**
Pruning process: **[E1,E2,....]**

Note
- The pruning process refers to all the pruning operations performed from the original table to the subtable.
- Give me the answer in format "Final Answer: True / False" form (should be either True or False, without any explanation)

Figure 19: Prompt of Table Verifier.

Textural Evidence

Question1:
how many cities in Tel Aviv or Center have under 65,000 people?
Evidence1:
area [colmn2:District] action [textural] condition [Center in x]
Question2:
how many competitions were not in the united kingdom?
Evidence2:
area [colmn4:Country] action [textural] condition [United Kingdom not in x]
Question3:
what is the total number of fw-190's he damaged or destroyed?
Evidence3:
area [colmn6:Status] action [textural] type [damaged in x]

Figure 20: Example of Textural Evidence.

Numerical Evidence

Question1:
how many cities in Tel Aviv or Center have under 65,000 people?
Evidence1:
area [colmn3:Population\n(2009)] action [numerical] condition [x < 65000]
Question2:
what is the total number of films made, with a rating of 8 or higher?
Evidence2:
area [colmn5:IMDb] action [numerical] condition [x >= 8]
Question3:
how many countries had at least $1 billion in box office?
Evidence3:
area [colmn3:Box Office] action [numerical] condition [x >= 1.0]

Figure 21: Example of Numberical Evidence.

Date Evidence

Question1:
how many games were played after october 1st?
Evidence1:
area [colmn1:Date] action [date] condition [x > 2008-10-01]
Question2:
what were the total number of deaths in 2003?
Evidence2:
area [colmn1:Year] action [date] condition [x == 2003-01-01]
Question3:
is the price money for 23 january 1984 more than that of 23 april 1984?
Evidence3:
area [colmn2:Date] action [date] condition [x == 1984-04-23]

Figure 22: Example of Date Evidence.

Error Case of ParserException

Question:
what was the highest score achieved by any opponent team?

Table:
col : Date | Team | Competition | Round | Leg | Opponent | Location | Score
row 1 : July 7 | Gent | Intertoto Cup | Round 2 | Leg 1, Home | Cliftonville | Jules Ottenstadion, Ghent | 2-0
row 2 : July 14 | ..... | 4-0
row 3 : July 21 | ..... | 1-1
row 4 : August 16 | ..... | 3-0
row 5 : March 6 | ..... | 0-5
row 6 : March 12 | ..... | 2-1

SQL:
SELECT MAX(CAST(SPLIT_PART(Score, '-', 2) AS INT)) AS MaxOpponentScore FROM table

Figure 23: Error Case of ParserException.

Error Case of BinderException

Question:
In cycle 4 of Austria's Next Top Model, how many contestants were older than 20?

Table:
col : Contestant | Age | Height | Home City | Rank
row 1 : Alina Chlebecek | 18 | 170cm | Vienna | Eliminated in Episode 1
row 2 : Isabelle Raisa | 16 | 170cm | Vienna | Eliminated in Episode 1
row 3 : Sabrina Angelika Rauch | 21| 170cm | Vienna | Eliminated in Episode 2
row 4 : Katharina Mihalovi | 23 | 170cm | Vienna | Eliminated in Episode 3
row 5 : Nata Mari | 16 | 175cm | Vienna | Eliminated in Episode 3
row 6 : Michaela Schopf | 21 | 172cm | Vienna | Eliminated in Episode 4
.....

SQL:
No such column
SELECT COUNT(*) FROM table WHERE Cycle = 4 AND CAST(Age AS INT) > 20

Figure 24: Error Case of BinderException.