

ALTER: Augmentation for Large-Table-Based Reasoning

Han Zhang¹², Yuheng Ma¹², Hanfang Yang^{12*}

¹Center for Applied Statistics, Renmin University of China

²School of Statistics, Renmin University of China

{hanzhang0816, yma, hyang}@ruc.edu.cn

Abstract

While extensive research has explored the use of large language models (LLMs) for table-based reasoning, most approaches struggle with scalability when applied to large tables. To maintain the superior comprehension abilities of LLMs in these scenarios, we introduce **ALTER** (Augmentation for Large Table-basEd Reasoning)-a framework designed to harness the latent augmentation potential in both free-form natural language (NL) questions, via the query augmentor, and semi-structured tabular data, through the table augmentor. By utilizing only a small subset of relevant data from the table and supplementing it with pre-augmented schema, semantic, and literal information, ALTER achieves outstanding performance on table-based reasoning benchmarks. We provide a detailed analysis of our method in large-table scenarios, comparing competitive baselines with various table partitioning principles. Our method **outperforms all other approaches** and exhibits robustness and efficiency against perturbations in all large-table scenarios. Our code is available at <https://github.com/Hanzhang-lang/ALTER>.

1 Introduction

Tabular data is one of the fundamental and critical semi-structured data types widely used in relational databases, spreadsheets, analysis reports, *etc.* Table-based reasoning tasks, such as table-based fact verification (FV) (Aly et al., 2021; Chen et al., 2020a; Ou and Liu, 2022) and table-based question answering (TQA) (Chen et al., 2020b; Pasupat and Liang, 2015; Lu et al., 2023; Cheng et al., 2022) require sophisticated reasoning over textual, numerical, and logical forms. Additionally, inference based on large-scale tables is in substantial practical demand and poses significant challenges for machine intelligence.

Recently, large language models (LLMs) have demonstrated remarkable proficiency in reasoning. The advent of LLMs has spurred a surge in research focusing on their application to tabular data, heralding what can be termed the **LLM era** (Zhang et al., 2024; Lu et al., 2024). Despite techniques following the **pre-LLM era**, such as fine-tuning methods, the latest LLM-based approaches have achieved results that are on par with or surpass those obtained through rule-based or pre-trained language model approaches (Liu et al., 2022; Gu et al., 2022; Jin et al., 2022), leveraging the contextual understanding capabilities of LLMs.

Mainstream techniques in the LLM era focus on designing prompts or pipelines that combine instructions with serialized natural language descriptions converted from tables, without additional training. The sequential text data is parsed by LLMs, transformed into executable code (*e.g.*, SQL and Python) using symbolic code generation abilities (Zan et al., 2023; Cheng et al., 2023) or direct output for inference utilizing literal reasoning abilities (Jiang et al., 2023; Gong et al., 2020).

However, most table-based methods encounter three challenges when analyzing complex large tables. **Firstly**, in the process of converting table cells into natural language descriptions, the entire data is often expected to be included to provide enough comprehensive information (Cheng et al., 2023). This approach can sometimes face data leakage issues involving privacy concerns and may fail due to context length limitations. In addition, the excessive length of all tabular content introduces unnecessary computational resource consumption and potential bias. **Secondly**, table reasoning tasks often require numerical reasoning, data preparation, or key cell identification. LLMs alone may lack the robustness to address these tasks directly and can sometimes introduce inaccuracies or hallucinations in their outputs. As tables grow in size, reasoning about minor or nuanced details becomes

*Corresponding author.

even more difficult (Liu et al., 2024), and LLMs require careful design to enhance their expandability and robustness in such scenarios. **Thirdly**, relevant parts needed to derive the answer may be scattered in different places for a complex large-table reasoning task. Therefore, intricate queries cannot be answered directly or resolved through a single step of program execution. Although a couple of methods have been optimized for specific issues mentioned above, no approach simultaneously considers all these problems while extending table-based reasoning tasks to large-scale tables.

In consideration of the issues mentioned above, we propose a novel framework named **ALTER** to facilitate the understanding of tables and to scale effectively to large tables. Instead of utilizing the **entire table data** as contextual information throughout the process, we maintain the contextual length by fixing the number of rows input into LLMs and selectively filtering the pertinent columns. We enhance table comprehension by leveraging various types of augmented information. The query augmentor generates adaptations about the NL questions and the table augmentor generates interpretations about the table’s inherent structure and content. The token length of these contents exhibits robustness to variations in table size. In conjunction with augmented information, the well-organized filtered data is integrated with SQL executors and ultimately transformed into a more accessible format for joint reasoning, adhering to the proposed *augment-filter-execute* procedure.

In summary, our main contributions include: (i) We explore new augmentation methods for queries and tables that enhance table reasoning tasks. (ii) We propose a general framework and a novel *augment-filter-execute* procedure capable of scaling to large tables. (iii) We conduct extensive experiments on table reasoning benchmarks, demonstrating superior performance and exhibiting robustness to perturbations in large-table scenarios.

2 Related Work

Large Language Models for Table Reasoning. Primary approaches using LLMs to tackle table reasoning tasks involve fine-tuning a foundational model or directly utilizing in-context learning abilities unique to the LLM era. For fine-tuning methods, task-specific fine-tuning methods are designed. TaPas (Herzig et al., 2020) extends BERT’s (Devlin et al., 2019) architecture and enhances the under-

standing of tabular data by recovering masked cells. Models relying on logical codes (e.g., SQL) can further enhance the model’s reasoning ability. For example, Tapex (Liu et al., 2022) and OmniTab (Jiang et al., 2022) focus on generating SQL queries that are then executed to fetch relevant information.

Prompting technologies such as few-shot learning (Brown et al., 2020a), chain-of-thought reasoning (COT) (Wei et al., 2022), and agent-based methods (Wang et al., 2024a) can be correspondingly applied in table reasoning tasks. Chen (2023) first explores and demonstrates the feasibility of using LLMs in generic reasoning tasks. Binder (Cheng et al., 2023) shows symbolic languages are also beneficial for complex analysis with prompt methods. Chain-of-Table (Wang et al., 2024b), inspired by CoT prompting methods, uses tabular data in the reasoning chain as a proxy for intermediate thoughts. ReAcTable (Zhang et al., 2023) employs LLMs extending the ReAct framework to reason step-by-step and iteratively generates sub-tables using code executors. Dater (Ye et al., 2023) and DIN-SQL (Pourreza and Rafiei, 2023) break down table reasoning into multi-step inference by hand-crafting pipeline.

Query Augmentation. In question-answering tasks, query augmentation or query rewriting is a prevalent method to bridge the gap between queries and facts. Within the framework of LLMs, tasks related to Retrieval-Augmented Generation (RAG) often involve various forms of query modification, including query rewriting, disambiguation, and decomposition (Gao et al., 2023). RQ-RAG (Chan et al., 2024) equips the model with multiple capabilities in multi-hop QA tasks. Ma et al. (2023) proposes *Rewrite-Retrieve-Read* pipeline which adapts the query itself. Step-Back Prompting (Zheng et al., 2024) presents a simple technique to derive high-level concepts. Our method further supplements sampled table content to better suit the table question answering scenario.

Table Augmentation and Table sampling. Table augmentation involves the exploration of implicit table content. Mainstream methods include the incorporation of commonsense knowledge from search engines (Sui et al., 2023) or analytical knowledge (He et al., 2023; Jena et al., 2022) into inference processes. Sui et al. (2024) leverages the LLM itself to augment structural information using internal knowledge. Instead, the augmentation in the table augmentor is closely aligned with our ALTER framework, which is utilized throughout

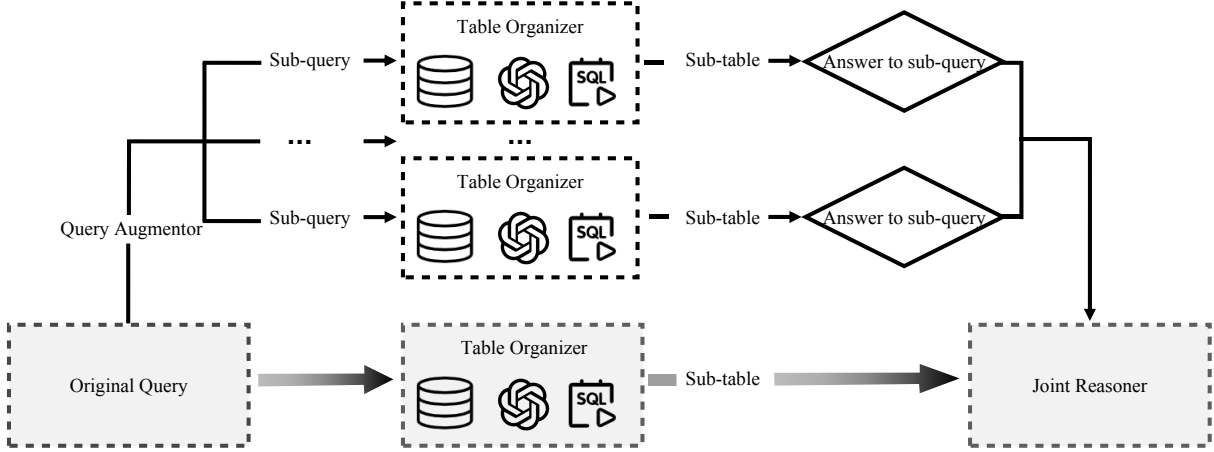


Figure 1: The overview of the ALTER framework for table-based reasoning. The gray background box symbolizes the primary reasoning workflow. Above it, each sub-query generated by the query augmentor is processed in parallel by the table organizer and ultimately transformed into informative demonstrations that aid in understanding the original query. The primary sub-table and relevant information is received by the joint reasoner.

the process. For table sampling, Lin et al. (2023) fine-tune DPR (Karpukhin et al., 2020) to retrieve sub-tables and TabSQLify (Nahid and Rafiei, 2024) relies on SQL queries to decompose tables into relevant sub-tables.

3 Preliminary

In this section, we introduce the definition of table reasoning tasks. Table reasoning requires reasoning over both free-form text and inherently structured tables. Given the triplet (T, Q, A) , where table $T = \{c_i\}_{i=1}^C$, C represents the number of column features in the table. Note that we do not represent the table in cell format as we expect the table under investigation to adhere to certain norms inherently. Q signifies a query or claim related to the table, and A denotes the answer.

We specifically focus on the table question answering and fact verification tasks. In the table question answering tasks, Q and A correspond to the query and expected answers in natural language form, respectively. In the table fact verification task, Q represents a claim about the table, and the final answer $A \in \{0, 1\}$ where 0 indicates falsity and 1 indicates truth regarding the input claim.

4 Methodology

4.1 Overview

In this work, we assume that semi-structured tabular data is rich in latent information beyond its raw data values. This information suggests that data storage adheres to certain common patterns or field semantics, facilitating the inference of the

overall data distribution from a minimal sample of data. Inspired by knowledge-fusion models for metadata inference (He et al., 2023) and the inherent knowledge-retrieving ability of LLMs (Sui et al., 2024), we utilize LLMs to uncover patterns and semantics within tables, which helps to understand and operate data correctly. The entire workflow is illustrated in Figure 1, with detailed steps outlined in Algorithm 1 in the appendix. In our framework, the full content of the table is not included in the prompt; only K sampled rows are observable. Nevertheless, the reasoning effect is ensured through the inclusion of elaborately augmented information. The framework seamlessly accommodates large-scale tables, as the model is pre-endowed with comprehensive information about the data structure and content prior to inference. As illustrated in Figure 1, our proposed system **ALTER**, consists of three core components:

- **Query Augmentor:** This component enhances the original query by generating multiple sub-queries, each examining the original query from different perspectives. Compared to the partial original query, this component comprehensively provides more information through the subsequent table organizer.

- **Table Organizer:** Given the input query, this component utilizes the *augment-filter-execute* procedure. It first guides LLMs to perform data mining, enriching the raw data with augmented table content, then filters the data to retain only highly relevant rows and columns, and finally employs an SQL executor to derive an accessible sub-table for

[Q] Which country had the most cyclists finish within the top 3?

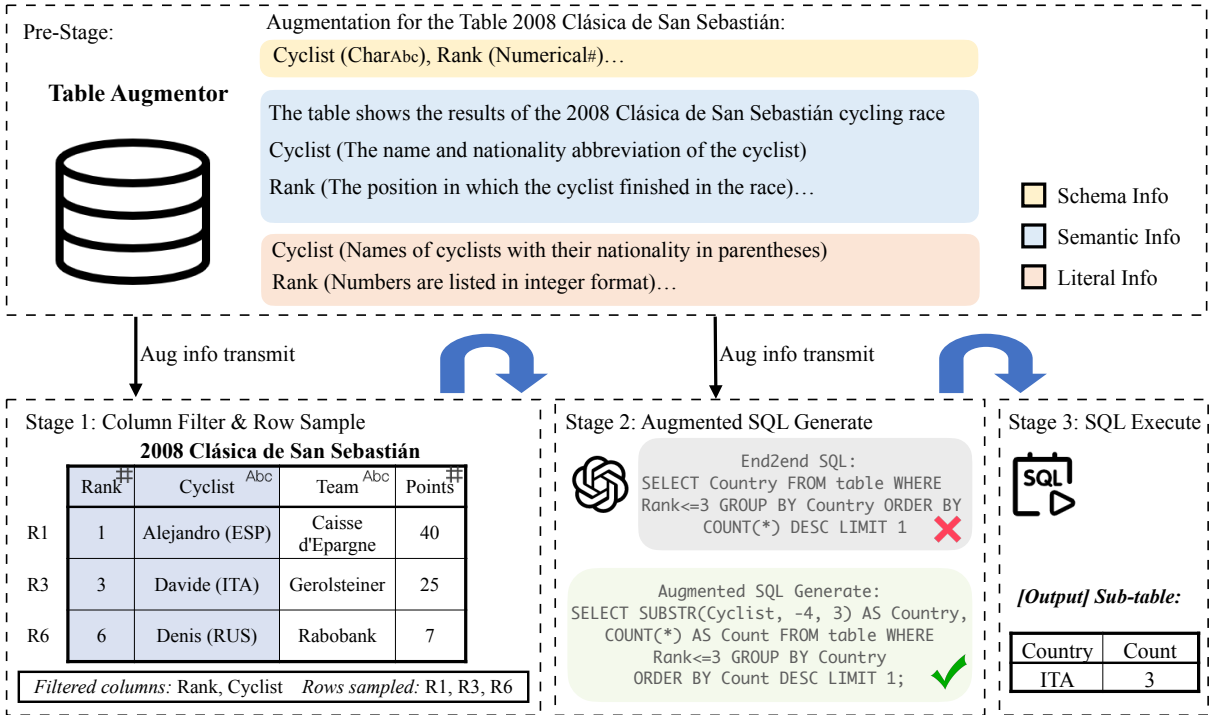


Figure 2: Illustration of the table organizer. The augmented information from the table augmentor is utilized in stage 1 and stage 2. In the example depicted in the figure, the model leverages the augmented information to accurately identify relevant columns and correctly parse nationalities within the table, ultimately producing the correct execution sub-table.

final inference.

- **Joint Reasoner:** This component efficiently performs reasoning and aggregation for the query augmentor and the primary workflow.

4.2 Query Augmentor

One of the primary challenges in naive Question Answering (QA) lies in its direct reliance on the user’s original query as the basis. In tabular reasoning scenarios, an imprudent query can lead to the model focusing on one partially biased part in the table. We propose a novel improvement method for the query part, which enables the LLMs to utilize the multi-query technique to address the original query from multiple perspectives. Each sub-query undergoes the reasoning process via the table organizer module, with this process being conducted in parallel. The model can utilize each independent reasoning module to attend to different parts within the table and extract information pertinent to answering the original query.

We propose two query augmentation methods: *step-back augmentation* and *sub-query augmentation*. The step-back prompting method (Zheng et al., 2024) has been empirically validated as ef-

fective in the RAG domain. We equip it with sampled sub-table information, which aims to obtain more abstract-level comprehension within the table through query rewriting. LLMs are shown to be stronger at sequentially solving sub-problems than directly solving a complex problem (Zhou et al., 2022a). The sub-query augmentation method decomposes complex queries into sub-queries, enabling LLMs to more easily locate the relevant information within each sub-query. Specifically, we leverage LLMs to generate distinct sub-queries based on the rewrite or decomposition demand. Detailed prompts for both augmentation methods are provided in Appendix F.

4.3 Table Organizer

The table organizer is the core component of the reasoning process. We do not use the entire table data as contextual information; instead, we further filter the column features of the table, as detailed in Section 4.3.2. To maintain model performance without accessing full data, we employ the *augment-filter-execute* strategy. By pre-analyzing the table’s schema, semantic, and literal information using LLMs, sufficient supplementary infor-

mation required by the query is provided. Notably, the augmented information does not increase commensurately with the table size. Therefore, our method can exhibit strong robustness to variations in table size.

The table organizer primarily encompasses one preparatory stage and three reasoning stages, as illustrated in Figure 2. During the preparatory stage, the table augmentor *augments* and stores enhanced information at various levels for subsequent processing. Schema information is utilized for standardizing the table content. In stage 1, semantic information is employed to identify relevant columns. Rows are sampled based on semantic similarity. In stage 2, with the *filtered* sub-table, the augmented information transmitted can be further simplified. We utilize literal and semantic information and leverage text-to-SQL capabilities of LLMs to generate high-quality SQL. The SQL query is *executed*, and the final sub-table is retrieved.

4.3.1 Table Augmentor

The table augmentor aims to convey extra information hidden inherently in the table and column features, beyond the raw data itself. The augmentation process occurs prior to the official reasoning process, as illustrated in Figure 2.

It’s worth noting that we can link this process to real large database systems or table applications (Xue et al., 2023). In standard database systems, extensive work on data cleaning and normalization must be undertaken. In real-world databases, column names are often represented by uppercase abbreviations or meaningless encrypted codes. The data stored may be formatted with abstract symbols, posing challenges in generating SQL queries accurately. Therefore, the table schemas typically require pre-defined, with the semantics of column features specified in advance. Hierarchical meta information will be synchronized, including information about the database, tables, and data stored. In such cases, we can simplify the steps of our table augmentor by migrating pre-defined augmented information.

In this paper, we leverage LLMs’ inherent knowledge extraction capabilities to augment table information based on the filtered sub-table. Specifically, we design three different augmentation types to suit the needs of downstream stages: schema information, semantic information, and literal representation. The prompt for each category of augmentation is detailed in Appendix F.

- Schema information primarily represents the data types of features stored in tables, which facilitates inferring and unifying data formats when reasoning over tables. We extracted three commonly used types in daily analysis: *Numerical*, *Char*, and *Date* types. These types are utilized to standardize and transform table data. Special symbols are pre-processed for *Numerical* and *Char* data, and different date representations are uniformly formatted for the *Date* type. The features ultimately stored in the database for SQL manipulation are transformed into corresponding data types.

- Semantic information primarily includes the global semantic information of the table and the semantics about column features. The global table information provide clues for identifying the relevant domain of the table. Utilizing feature-specific semantic information, LLMs can more accurately locate features related to the query. Specifically, when columns are named using acronyms or aliases, the imparted semantics can be pivotal for analysis. The semantic information is transmitted for column filtering in stage 1 and augmented SQL generation in stage 2.

- SQL queries often fail to accurately parse the correct format stored in the table. Chain-of-Table (Wang et al., 2024b) improves this by using multiple chain calls. However, by explicitly informing LLMs about the raw data representation format within the table through literal information, the model performs better in the generation of correctly formatted SQL queries in a single LLM call. Unlike semantic information, literal information focuses on the representation format of the data (*e.g.*, extra parentheses, calculation formulas, special expressions), which is efficient for SQL generation.

4.3.2 Column Filter and Row Sample

Irrelevant table content in the prompt can lead to unnecessary computations and quality regression issues (Sui et al., 2023), especially in scenarios involving large tables. We retain a small number of columns and rows from the original table. Unless otherwise specified, we set $K = 3$ in this paper, meaning the model can only access three rows of data relevant to the question throughout the process. However, through the table augmentor, we can obtain globally enhanced table information. Specifically, we first store the index of the vector representation of each row content, and search for K rows based on embedding-based semantic similarity between each row and the utterance. Sub-

sequently, a powerful LLM is utilized to select columns relevant to the query, excluding irrelevant ones. The prompt for column filtering is detailed in Appendix F. During the column pick, the augmented information is also used for comprehensive understanding. This module ensures that the scale of the sub-table passed to LLMs remains consistent regardless of the size of the original table.

4.4 Joint Reasoner

The Joint Reasoner is responsible for integrating upstream information to perform the final reasoning. To avoid interference from entirely irrelevant information, sub-queries that cannot be answered are discarded. Valid sub-queries are transformed into effective descriptions. These demonstrations are combined with the sub-table from the primary workflow to collectively aid in answering the original query. Please refer to Appendix F for further details and more comprehensive prompts. We leverage step-by-step reasoning capabilities of LLMs to arrive at the final answer.

5 Experiment

In this section, we first introduce the datasets and evaluation metrics. We compare ALTER with the baseline methods and present the results in Sections 5.2 and 5.3. The ablation study and analysis of large-table scenarios are discussed in Sections 5.4 and 5.5, respectively. Additional implementation details are provided in Appendix A.

5.1 Datasets and Evaluation Metrics

We evaluate our proposed method on two widely-used table-based reasoning benchmarks, WikiTQ (Pasupat and Liang, 2015) and TabFact (Chen et al., 2020a). For the table-based fact verification task, we adopt the TabFact dataset, which contains various statements based on Wikipedia tables. We evaluate the dataset using binary classification accuracy on the small-test set containing 1998 statements with 298 different tables.

For the table reasoning task, we adopt WikiTable-Question (WikiTQ), which contains open-domain tables accompanied by complex questions. We use denotation accuracy as our evaluation metric, which evaluates the predicted answers based on the gold ones. We evaluate our method on the test set containing 4344 samples from 421 different tables.

Table 1: Results of different methods on WikiTQ and TabFact.¹ (We use underline to denote the second-best performance, **bold** to denote the best performance for each region: Pre-LLM era, LLM era with result ensemble and without ensemble)

Method	Acc (%)	
	WIKITQ	TABFACT
♥ <i>Pre-LLM era</i>		
TAPEX (Liu et al., 2022)	57.2	85.9
TaCube (Zhou et al., 2022b)	60.8	-
ReasTAP (Zhao et al., 2022)	58.6	<u>86.2</u>
OmniTab (Jiang et al., 2022)	<u>62.7</u>	-
CABINET (Patnaik et al., 2024)	69.1	-
PASTA (Gu et al., 2022)	-	90.8
♠ <i>LLM era</i>		
Binder (Cheng et al., 2023)	55.1	85.1
Dater w SC (Ye et al., 2023)	69.0	85.4
ReAcTable w s-vote (Zhang et al., 2023)	68.0	86.1
Mix SC w SC (Liu et al., 2023)	73.7	-
Chain-of-Table (Wang et al., 2024b)	67.3	86.6
ALTER (ours) w SC	<u>70.7</u>	87.2
Dater w/o sc (Ye et al., 2023)	65.0	<u>83.5</u>
ReAcTable (Zhang et al., 2023)	<u>65.8</u>	83.1
Mix SC w/o SC (Liu et al., 2023)	64.2	-
ALTER (ours) w/o SC	67.4	84.3

5.2 Baselines

We compare the proposed ALTER with a range of advanced reasoning frameworks for table-based tasks. The baseline methods for comparison can be categorized into two types: mainstream techniques following the pre-LLM era and techniques unique to the LLM era. For the techniques following the pre-LLM era, we select TAPEX (Liu et al., 2022), ReasTAP (Zhao et al., 2022), TaCube (Zhou et al., 2022b), OmniTab (Jiang et al., 2022), CABINET (Patnaik et al., 2024). For the techniques unique to the LLM era, we select Binder (Cheng et al., 2023), Dater (Ye et al., 2023), ReAcTable (Zhang et al., 2023), Mix SC (Liu et al., 2023), Chain-of-Table (Wang et al., 2024b). Additionally, generating multiple reasoning paths and ultimately choosing the most consistent answer through voting or self-consistency (Wang et al., 2022) can enhance the performance of LLMs. Therefore, for the techniques unique to the LLM era, we report two types of results for those methods employing result ensemble techniques.

5.3 Results

We present the results on the WikiTQ and TabFact datasets. The experimental outcomes are summarized in Table 1. From the results, we observe

¹For the Dater method, we report the results of using the LLM-based method as backbone

Table 2: Ablation results of query augmentor on the test sets of WikiTQ and TabFact.

Methods	TABFACT			WIKITQ		
	All	Simple	Hard	All	Simple	Hard
ALTER	84.3	90.7	78.2	67.4	71.2	63.4
w/o <i>step-back</i>	82.3 (↓ 2.0)	89.5 (↓ 0.9)	75.4 (↓ 2.8)	64.5 (↓ 2.9)	68.2 (↓ 3.0)	60.5 (↓ 2.9)
w/o <i>sub-query</i>	82.4 (↓ 1.9)	90.6 (↓ 0.1)	74.6 (↓ 3.6)	65.4 (↓ 2.0)	69.7 (↓ 1.5)	60.8 (↓ 2.6)

that ALTER method achieves comparatively outstanding outcomes. Specifically, on the WikiTQ dataset, while the Mix SC method do marginally outperforms our method by aggregating multiple reasoning paths (with 10 sampling times), ALTER still managed to exceed the performance of all other methods under comparison. Notably, ALTER demonstrates the best performance in single-round reasoning among all other methods that utilize result ensemble techniques in the LLM era. This demonstrates the robust performance of our method in reasoning tasks, which can be attributed to the reinforced information provided by the query augmentor and our innovative modular procedure within the table organizer.

5.4 Ablation Study

We carry out an ablation study to assess the impact of various components on the performance of our methods, as well as to explore the relationship between the pure table data and the inherent augmentation information.

Analysis of the Query Augmentor. To analyze the impact of two query augmentation methods in the query augmentor. We conducted experiments on two datasets by discarding the step-back augmentation module (denoted as w/o step-back) and the sub-query augmentation module (denoted as w/o sub-query). For each dataset, we further categorized the questions based on the difficulty level, following Ye et al. (2023). This stratification facilitates a more comprehensive evaluation of each module’s impact across different types of questions. The ablation test results are reported in Table 2. From the results in the table, it is anticipated that employing both augmentation methods simultaneously yields the best performance under all experimental settings. For WikiTQ datasets, the accuracy of ALTER without step-back/sub-query augmentation drops by 2.9%/2.0%, demonstrating the necessity of augmented information from multi-queries. Furthermore, on the TabFact datasets, both augmentation methods have a much larger impact on hard questions than on simple questions. This

indicates that the augmented information provided by the query augmentor is particularly effective in dealing with complex questions.

Table 3: Ablation results of different values of rows sampled, *i.e.*, K and with or without augmented information on the WikiTQ and TabFact. (improvement measured against the data relative to the position on the bottom-left.)

	WIKITQ		TABFACT	
	w/o aug	w/ aug	w/o aug	w/ aug
$K = 0$	45.5	62.2	67.1	77.2
$K = 1$	59.2	65.0 (+1.7)	80.5	82.4 (+0.5)
$K = 3$	63.3	67.4	81.9	84.3

Analysis of Pure Data & Augmentation. In our experiments, we primarily utilized $K = 3$ rows of data as contextual information. To explore the relationship between pure table data and the augmented information in the table organizer, we conducted ablation experiments varying the value of K and the augmentation process. Results are shown in Table 3. We observe that methods utilizing augmented information exhibit significant performance improvements compared to those without augmented information. We also note that the concurrent absence of augmented information and data provision leads to a catastrophic decline in model performance. Notably, on both datasets, using only one row of data with augmented information achieves comparable performance to using three rows of data. Similar trends can also be observed in other settings. This validates that when the model is limited to a small portion of data, the table augmentor serves as a beneficial auxiliary tool, providing additional insights into the table’s content.

5.5 Large Table Analysis

LLMs often struggle to interpret tables within large-scale scenarios, leading to hallucinations and errors. To the best of our knowledge, nearly all methods encounter a decline in model performance as the table size increases when handling large tables.

Comparison under Large Table Scenarios. To demonstrate the effectiveness of the ALTER frame-

Table 4: Comparison of methods in the LLM era with tables divided by token count on WikiTQ. (underline denotes the second-best performance; **bold** denotes the best performance)

Methods	TABLE SIZE		
	Small (<2k)	Medium (2k~4k)	Large (>4k)
Binder	56.5	26.1	6.4
Dater	62.5	42.3	34.6
Chain-of-Table	<u>68.1</u>	<u>52.3</u>	<u>44.9</u>
ALTER (ours)	71.7 (+3.6)	65.2 (+12.9)	65.9 (+21.0)

work in large-scale scenarios, we compare the performance of our framework across different table sizes in this section. We selected various table partitioning principles and different types of methods for a systematic evaluation. For table partitioning, we employed two approaches based on the token count and the number of cells. For the models, representative methods from both the LLM era and the pre-LLM era are chosen.

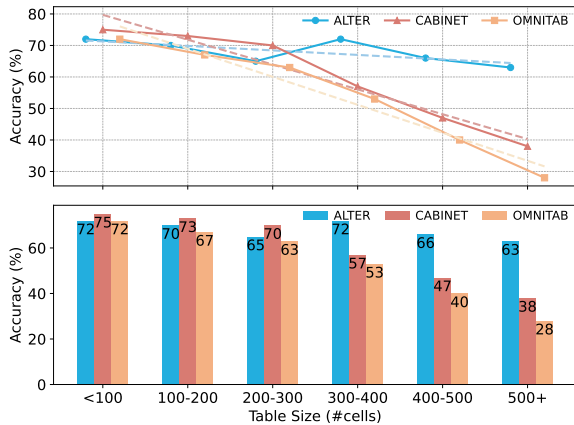


Figure 3: Comparison of methods following pre-LLM era with tables divided by cell count on WikiTQ. In the subplot above, the regression curves of different models are represented by dashed lines in different colors. The regression curve for ALTER exhibits a significantly slower decline rate.

Figure 3 shows the comparison results of ALTER and methods following the pre-LLM era, including CABINET and OMNITAB, partitioning tables in the WikiTQ dataset by the number of cells. In Table 4, we present the results based on different table sizes divided by the token count in the WikiTQ dataset, comparing our method with Dater, Chain-of-TABLE, and Binder unique to the LLM era. Table 4 shows that ALTER significantly outperforms all three methods in the LLM era across different table sizes. The performance improvement is particularly noteworthy when dealing with large tables. In Figure 3, our model demonstrates a much slower performance decline as the model

size increases compared to the other two methods. As the size of the table increases, both CABINET and OMNITAB exhibit a monotonous decline in performance. However, our method shows a brief reversal with an increase in performance observed in the intermediate range, indicating the robustness and insensitivity of our approach to changes in table size. Our model significantly outperforms the other two methods when the table size exceeds a certain threshold (> 300 cells). Specifically, in the 300 – 400, 400 – 500, and 500+ cell categories, our model exceeds their performance by at least 15%, 19%, and 25%, respectively. From the results, it is evident that our method exhibits exceptional performance in large tables.

Robustness and Efficiency Analysis. We examined ALTER’s robustness to noise perturbations and token efficiency in large-scale scenarios. By adding random rows based on different perturbation factors, we introduced noise to each table in WikiTQ, details of perturbations can be found in Appendix E. From Figure 4, we illustrate that as the degree of perturbation increases, the proportion of tokens utilized of the whole table by ALTER decreases. It can be observed that the initial fluctuation has the most significant effect, yet our model still outperforms the compared method (9.8% ALTER *v.s.* 11.4% CABINET). Concurrently, the decline in the framework’s performance degree slows down. This indicates that our method efficiently maintains robust performance in large-table scenarios by narrowing down the scope of larger tables.

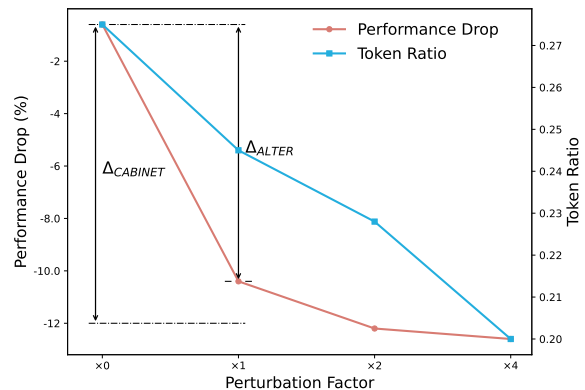


Figure 4: Relative performance drop and the ratio drop of the table tokens utilized by ALTER to the total token count over the table as the number of rows added increases by multiples (*i.e.*, perturbation factor) on WikiTQ. Specifically, the performance drop for CABINET and ALTER is marked at the factor of 1.

5.6 Case Study

In Appendix B, we present a case study illustrating how each component of the augmented information in ALTER framework contributes to accurate comprehension or leads to errors. When addressing complex problems, without the assistance of the augmentation process, the model may focus on biased information or experience hallucinations when generating SQL. However, when the augmented information is explicitly provided, the model can identify the region containing the correct information or generate syntactically correct SQL, thereby delivering accurate responses.

6 Conclusion

We propose a framework, namely ALTER, which significantly optimizes model performance on large-scale tables. Within this framework, we extract inherent information pertinent to the questions and tables. By leveraging an *augment-filter-execute* process as the core reasoning workflow, ALTER demonstrates superior performance in handling large tables. We believe ALTER can bridge the gap between table reasoning methodologies and real-world analysis and bring insights into understanding the way LLMs comprehend tables.

Limitations

ALTER is designed to generalize to large table reasoning tasks, but our method still faces some limitations. Our approach relies partly on the degree of structured and standardized storage of tables, meaning that if the table structure is totally disordered or lacks a certain level of standardization, our model’s performance will degrade, for instance, when headers and data are intermixed. Additionally, the combination methods of different augmented information can be explored further. Due to the page limits, we will leave these explorations for future work.

Acknowledgment

Hanfang Yang is the corresponding author. The authors would like to thank the reviewers for their constructive comments, which led to a significant improvement in this work. The research is supported by the Special Funds of the National Natural Science Foundation of China (Grant No. 72342010). Yuheng Ma is supported by the Outstanding Innovative Talents Cultivation Funded Programs 2024 of Renmin University of China. This research is also

supported by Public Computing Cloud, Renmin University of China.

References

- Rami Aly, Zhijiang Guo, Michael Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. Feverous: Fact extraction and verification over unstructured and structured information. *arXiv preprint arXiv:2106.05707*.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020a. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020b. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS ’20*, Red Hook, NY, USA. Curran Associates Inc.
- Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. 2024. RQ-RAG: Learning to refine queries for retrieval augmented generation. *arXiv preprint arXiv:2404.00610*.
- Wenhu Chen. 2023. [Large language models are few\(1\)-shot table reasoners](#). In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1120–1130, Dubrovnik, Croatia. Association for Computational Linguistics.
- Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020a. Tabfact: A large-scale dataset for table-based fact verification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.
- Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020b. [HybridQA: A Dataset of Multi-Hop Question Answering over Tabular and Textual Data](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1026–1036, Online. Association for Computational Linguistics.
- 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.
- 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**. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. **BERT: Pre-training of deep bidirectional transformers for language understanding**. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.
- Heng Gong, Yawei Sun, Xiaocheng Feng, Bing Qin, Wei Bi, Xiaojiang Liu, and Ting Liu. 2020. **Tablegpt: Few-shot table-to-text generation with table structure reconstruction and content matching**. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1978–1988, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- 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**. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4971–4983, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Xinyi He, Mengyu Zhou, Mingjie Zhou, Jialiang Xu, Xiao Lv, Tianle Li, Yijia Shao, Shi Han, Zejian Yuan, and Dongmei Zhang. 2023. Anameta: A table understanding dataset of field metadata knowledge shared by multi-dimensional data analysis tasks. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 9471–9492.
- Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisen-schlos. 2020. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4320–4333.
- Aashna Jena, Vivek Gupta, Manish Shrivastava, and Julian Eisen-schlos. 2022. **Leveraging data recasting to enhance tabular reasoning**. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4483–4496, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. **StructGPT: A general framework for large language model to reason over structured data**. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, Singapore. 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.
- Nengzheng Jin, Joanna Siebert, Dongfang Li, and Qingcai Chen. 2022. A survey on table question answering: recent advances. In *China Conference on Knowledge Graph and Semantic Computing*, pages 174–186. Springer.
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. **Dense passage retrieval for open-domain question answering**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.
- 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.
- Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.
- Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022. TAPEX: table pre-training via learning a neural SQL executor. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*.
- Tianyang Liu, Fei Wang, and Muhao Chen. 2023. Rethinking tabular data understanding with large language models. *arXiv preprint arXiv:2312.16702*.

- Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2023. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.
- Weizheng Lu, Jiaming Zhang, Jing Zhang, and Yueguo Chen. 2024. Large language model for table processing: A survey. *arXiv preprint arXiv:2402.05121*.
- Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. [Query rewriting in retrieval-augmented large language models](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315, Singapore. Association for Computational Linguistics.
- Md Nahid and Davood Rafiei. 2024. [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, Mexico City, Mexico. Association for Computational Linguistics.
- Suixin Ou and Yongmei Liu. 2022. [Learning to generate programs for table fact verification via structure-aware semantic parsing](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7624–7638, Dublin, Ireland. Association for Computational Linguistics.
- 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 of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1470–1480.
- Sohan Patnaik, Heril Changwal, Milan Aggarwal, Sumit Bhatia, Yaman Kumar, and Balaji Krishnamurthy. 2024. CABINET: Content relevance-based noise reduction for table question answering. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*.
- Mohammadreza Pourreza and Davood Rafiei. 2023. [DIN-SQL: Decomposed in-context learning of text-to-sql with self-correction](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 36339–36348. Curran Associates, Inc.
- 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, WSDM '24*, page 645–654, New York, NY, USA. Association for Computing Machinery.
- Yuan Sui, Jiaru Zou, Mengyu Zhou, Xinyi He, Lun Du, Shi Han, and Dongmei Zhang. 2023. [Tap4llm: Table provider on sampling, augmenting, and packing semi-structured data for large language model reasoning](#). *arXiv preprint arXiv:2312.09039*.
- Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024a. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):1–26.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.
- 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. 2024b. [Chain-of-table: Evolving tables in the reasoning chain for table understanding](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. [C-pack: Packaged resources to advance general chinese embedding](#). *Preprint, arXiv:2309.07597*.
- Siqiao Xue, Caigao Jiang, Wenhui Shi, Fangyin Cheng, Keting Chen, Hongjun Yang, Zhiping Zhang, Jianshan He, Hongyang Zhang, Ganglin Wei, et al. 2023. [DB-GPT: Empowering database interactions with private large language models](#). *arXiv preprint arXiv:2312.17449*.
- 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.
- Daoguang Zan, Bei Chen, Fengji Zhang, Dianjie Lu, Bingchao Wu, Bei Guan, Wang Yongji, and Jian-Guang Lou. 2023. [Large language models meet NL2Code: A survey](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7443–7464, Toronto, Canada. Association for Computational Linguistics.

- Xuanliang Zhang, Dingzirui Wang, Longxu Dou, Qingfu Zhu, and Wanxiang Che. 2024. A survey of table reasoning with large language models. *arXiv preprint arXiv:2402.08259*.
- Yunjia Zhang, Jordan Henkel, Avriella Floratou, Joyce Cahoon, Shaleen Deep, and Jignesh M Patel. 2023. Reactable: Enhancing react for table question answering. *arXiv preprint arXiv:2310.00815*.
- Yilun Zhao, Linyong Nan, Zhenting Qi, Rui Zhang, and Dragomir Radev. 2022. Reastap: Injecting table reasoning skills during pre-training via synthetic reasoning examples. In *2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*.
- Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. 2024. [Take a step back: Evoking reasoning via abstraction in large language models](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*.
- Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022a. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.
- Fan Zhou, Mengkang Hu, Haoyu Dong, Zhoujun Cheng, Fan Cheng, Shi Han, and Dongmei Zhang. 2022b. Tacube: Pre-computing data cubes for answering numerical-reasoning questions over tabular data. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2278–2291.

A Implementation Details

All experiments in this paper were conducted on GPU clusters with 4 NVIDIA A100 GPUs. We employ GPT-3.5-turbo as our large language model backbone for all experiments. To ensure consistent results, we apply a self-consistency technique with 5 sampling times for each benchmark dataset. For the embedding model for column filter in Section 4.3.2, we utilize bge-large-en model (Xiao et al., 2023) and employ FAISS (Johnson et al., 2019) for efficient similarity search.

B Case Study

In Figure 5, the input question asks for the *vehicle preceding the Jaguar XJS*. When filtered table is directly provided, the SQL only attends to the second last row of the table. This indicates that the model has observed biased data, incorrectly assuming that the vehicle Jaguar XJS appears only once. However, through step-back query augmentation, the query is reframed, and the model generates a more general SQL query, acquiring more results and thus arriving at the correct answer.

In Figure 6, the query seeks to determine the tenure of *René Heitmann* as head coach. This involves operations on two distinct columns. By decomposing the original query into sub-queries, the difficulty is reduced, allowing the model to accurately retrieve the corresponding information and ultimately compute the correct result. In Figure 7, the input query seeks to determine the score differential for the team *Detroit*. Without relying on the augmented information from the table augmentor, the model fails to correctly capture the name in the *Team* column and cannot accurately extract the score values in the *Score* column. After incorporating the augmented information, the model generates syntactically correct SQL and extract the needed data.

We additionally present cases when the table augmentor and query augmentor brings in errors. In Figure 8, an erroneous response generated by the step-back query augmentor adversely affects the final query result. In Figure 9, the failure to generate correctly formatted augmentation information results in the loss of year column features. The incorporation of augmentation information enables the model to generate the correct SQL, underscoring the necessity of augmentation information.

1978 Trans-Am season

row_number	Date	Circuit	Winning_driver TA1	Winning_vehicle TA1
1	May 21	Sears Point	Gene Bothello	Chevrolet Corvette
2	June 4	Westwood	Nick Engels	Chevrolet Corvette
3	June 11	Portland	Bob Matkowitch	Chevrolet Corvette
4	June 25	Mont-Tremblant	Bob Tullius	Jaguar XJS
5	July 8	Watkins Glen	Brian Fuerstenau Bob Tullius	Jaguar XJS
6	August 13	Brainerd	Bob Tullius	Jaguar XJS
7	August 19	Mosport	Bob Tullius	Jaguar XJS
8	September 4	Road America	Bob Tullius	Jaguar XJS
9	October 8	Laguna Seca	Bob Tullius	Jaguar XJS
10	November 5	Mexico City	Bob Tullius	Jaguar XJS

Example query: which ta1 vehicle won previous to the jaguar xjs?

ALTER w/o query augmentor

```
SQL:
SELECT Winning_vehicle_TA1 FROM DF WHERE
row_number = (SELECT MAX(row_number) FROM DF WHERE
Winning_vehicle_TA1 = 'Jaguar XJS') - 1;
```

Final Answer: *Jaguar XJS*

✗

Step-back query augmentation:

which vehicles won in the TA1 category before the Jaguar XJS?

```
SQL:
SELECT Winning_vehicle_TA1 FROM table WHERE row_number < 7;
```

Final Answer: *Chevrolet Corvette*

✓

Figure 5: Intuitive example for step-back query augmentation, where ALTER correctly answers the query utilizing broader information compared to directly output SQL based on the original query.

C Error Analysis

We systematically examine the error patterns of ALTER. We randomly sampled 100 error cases from the WikiTQ dataset and manually analyzed these errors. The errors were subsequently categorized into six distinct types. Table 5 summarizes the predominant error types. According to the error distributions, ALTER may encounter failures due to the limited possessed data. The issue can be effectively mitigated by increasing the value of K . However, expanding the sample size incurs higher costs. In practice, it is essential to balance computational overhead and retrieval quality, achieving trade-off by selecting an optimal K value.

D Impact Analysis of the Table Augmentor

Within the ALTER framework, the integration of three types of augmentation information in the table augmentor is pivotal. For instance, schema information is employed for the pre-normalization of tables, while both table semantic and literal information are utilized concurrently during the generation of SQL queries. Consequently, isolating the impact of a single type of augmented information is complex. Nonetheless, in this section, we con-

Table 5: Proportions of different error types on WikiTQ.

Error Types	Ratio	Description
Hallucination	34%	LLMs incorrectly interpret the table content with hallucination.
Coding Errors	20%	LLMs produce inaccurate code, mainly due to minor format errors.
Selection Error	9%	Incorrect columns were selected in the column filter; sampling (K) rows resulted in biased data, leading to bias or incompleteness in the augmentation phase
Jointly Reasoner Error	16%	LLMs exhibit failures in jointly reasoner; contradictions or incorrect format, <i>e.g.</i> , 1 minute and 46.7 seconds or 1:46.07
Schema error	10%	Errors or biases occur during schema normalization
Other Errors	11%	Other uncategorizable errors

Table 6: Ablation results of table augmentor on the test sets of WikiTQ and TabFact.

Methods	WIKITQ	TABFACT
ALTER	67.4	84.3
w/o semantic information	64.8 (↓ 2.6)	83.4 (↓ 0.9)
w/o literal information	63.9 (↓ 3.5)	82.2 (↓ 2.1)

duct an ablation study evaluating the performance on the WikiTQ and TabFact datasets without the use of either semantic or literal information through the reasoning process. The experiments revealed that omitting either type of information leads to a decline in performance. Overall, a practical experience involves enriching the table augmentor module with additional types of information, such as the structural orientation of the table (horizontal or vertical), web knowledge, *etc.*, rather than relying solely on a single type of augmentation.

E Details of Table Perturbation

We provide details of the perturbations implemented in Section 5.5. We insert noise by adding rows based on the size of the table, following the row adding steps in Patnaik et al. (2024). However, we do not randomly extract values from other tables, as this would compromise the pre-augmented schema standardization. Based on the augmented schema information, we randomly generated data for three types of features: *Date*, *Numerical*, and *Char*. We believe the disturbance intensity is quite similar for the model compared to the previous approach. Based on the number of cells ($\#cells = N$) in the table, the exact scheme of the n rows inserted is as follows: (i) $n = 1$ if $N \leq 150$, (ii) $n = 2$ if $150 < N \leq 300$, (iii) $n = 4$ if $300 < N \leq 450$, (iv) $n = 8$ if $N \geq$

450. Additionally, for each of these categories, we vary the degrees of perturbation by multiplying the number of added rows by 1, 2, and 4 times (*i.e.*, perturbation factor used in Figure 4).

F Prompts in ALTER

We provide the prompt templates for each module used within the ALTER framework. In these templates, the red text serves as a placeholder for specific input, the blue text stands for a special placeholder for specific serialized table. In our work, the sub-tables are serialized into HTML format throughout the experiments following the practical guide in Sui et al. (2024). For a clear demonstration, we illustrate one demo for serialized table in Figure 10. The in-context few-shot examples are selected from the training or validation set for each task.

Prompts for the Query Augmentor. For each augmentation method in the query augmentor, we use 3 prompting examples. The detailed prompt is illustrated in Figure 11. Low-quality sub-queries are rejected by the joint reasoner module, and duplicate sub-queries are filtered out. This does not introduce excessive consumption, as the sub-table contains sufficiently few contents throughout the process, and each sub-query reasoning is conducted in parallel.

Prompts for the Table Organizer. In the table organizer, we show the prompt for each augmentation type in Figure 12 and the detailed prompt for the remaining module in Figure 13. For the column filter, we primarily use the global semantic information of the table and its feature semantics. In the SQL generation part, we focus on the global table semantics and the semantic and literal information of the filtered columns. In our experiments, We use 3 prompting examples in the column filter. We found that excessive inclusion of manually designed examples can adversely affect the predictive quality of the model, as also shown by Brown et al. (2020b). We deploy the augmented SQL Generation process in a zero-shot manner, which provides versatility in SQL generation and flexibility for adjusting to other text-to-SQL models as well.

Prompts for the Joint Reasoner. Before entering the joint reasoner, sub-queries are transformed into effective auxiliary descriptions. We provide the prompt for the joint reasoner, as well as the prompt for reasoning over sub-queries from the query augmentor to obtain extra information in Figure 14.

Boldklubben Frem

Name	Nationality	c_From	To	Comments
Henrik Jensen	Denmark	1 July 2012	Present	
John 'Tune' Kristiansen	Denmark	18 June 2012	23 June 2012	Caretaker for ...
Peer F. Hansen	Denmark	1 January 2012	18 June 2012	
John 'Tune' Kristiansen	Denmark	27 July 2010	3 December 2011	Originally had contract ...
René Heitmann	Denmark	17 July 2010	27 July 2010	Never coached the team ...
Christian Andersen	Denmark	11 July 2009	19 June 2010	Club went bankrupt ...
Anders Theil	Denmark	7 November 2010	7 July 2009	Originally had contract until summer 2011

Example query: how long was René Heitmann the head coach of Boldklubben Frem?

ALTER w/o query augmentor

SQL:
 SELECT * FROM DF WHERE Name = 'René Heitmann'
 AND Comments LIKE '%head coach%';

Final Answer: *No data from database* ❌

Sub-query query augmentation:

- when did René Heitmann start as head coach of Boldklubben Frem?
- when did René Heitmann stop being head coach of Boldklubben Frem?

SQL:
 SELECT c_From FROM DF WHERE Name = 'René Heitmann';
 SELECT To FROM DF WHERE Name = 'René Heitmann';

Final Answer: *10 days* ✅

Figure 6: Intuitive example for sub-query query augmentation, where ALTER correctly answers the query utilizing sub-queries compared to directly output SQL based on the original query.

2008-09 Sacramento Kings season

Date	Team	Score	High_points	Record
January 2	@ Detroit	L 92-98	Brad Miller (25)	8-25
January 3	@ Indiana	L 117-122	Kevin Martin (45)	8-26
January 5	@ New Jersey	L 90-98	Kevin Martin (36)	8-27
January 6	@ Chicago	L 94-99	Kevin Martin (29)	8-28
January 14	@ Golden State	W 135-133 (3OT)	Brad Miller (30)	10-30
January 20	@ Denver	L 99-118	Kevin Martin (25)	10-32
January 24	@ Milwaukee	L 104-106	Kevin Martin (20)	10-34

Example query: what was the point difference between the Sacramento and Detroit for game 33?

ALTER w/o table augmentor

SQL:
 SELECT ABS(CAST(SUBSTR(Score, 3) AS INTEGER)
 - CAST(SUBSTR(Score, 6) AS INTEGER)) AS Point_Difference
 FROM DF
 WHERE Team = 'Detroit' AND Game = 33;

Final Answer: *No data from database* ❌

Table organizer augmentation:

Team: Team names with @ indicating away games
 Score: Scores in the format W/L followed by points in the format of x-y for each team

SQL:
 SELECT ABS(SUBSTR(Score, INSTR(Score, '-') + 1, 2) -
 SUBSTR(Score, INSTR(Score, ' ') + 1, 2))
 AS Point_Difference FROM DF
 WHERE Team = '@ Detroit' AND Game = 33;

Final Answer: *6* ✅

Figure 7: Intuitive example for table augmentor, where ALTER correctly answers the query utilizing information about data format and composition compared to directly output SQL without any augmentation information.

1998 Sydney to Hobart Yacht Race

Position	Yacht	State_Country	Skipper	Elapsed_Time
1	Sayonara	USA	Larry Ellison	2:19:03:32
2	Brindabella	NSW	George Snow	2:21:55:06
3	Ausmaid	SA	Kevan Pearce	3:06:02:29
4	Ragamuffin	NSW	Syd Fischer	3:06:11:29
5	Nokia	CI	David Witt	3:09:19:00
6	Fudge	VIC	Peter Hansen	3:11:00:26
7	Quest	NSW	Bob Steel	3:14:41:28
8	Industrial Quest	QLD	Kevin Miller	3:14:58:46

Example query: what yacht had the next best time (smaller time is better) than Ausmaid?

Step-back query from query augmentor:
 which yacht had a better time than Ausmaid?

SQL:
 SELECT Yacht FROM DF WHERE Elapsed_Time_nd_hh_mm_ss <
 '3:06:02:29' AND Yacht != 'Ausmaid';

Answer from sub-query: the yacht that had a better time than Ausmaid is *Sayonara*

SQL in the primary workflow:
 SELECT Yacht FROM DF ORDER BY Elapsed_Time_nd_hh_mm_ss
 LIMIT 1 OFFSET 1;

SQL result: *Brindabella*

Final Answer: *Sayonara* ❌

Figure 8: Error case when the query augmentor generates biased demonstrations.

Škoda Auto

Model	c_2000	c_2001	c_2002	c_2003	c_2004	c_2005	c_2006
Škoda Felicia	148,028	44,963	-	-	-	-	-
Škoda Octavia	158,503	164,134	164,017	165,635	181,683	233,322	270,274
Škoda Fabia	128,872	250,978	264,641	260,988	247,600	236,698	243,982
Škoda Superb	-	177	16,867	23,135	22,392	22,091	20,989
Škoda Roomster	-	-	-	-	-	-	14,422
Škoda Yeti	-	-	-	-	-	-	-
Škoda Rapid	-	-	-	-	-	-	-
Škoda Citigo	-	-	-	-	-	-	-

Example query: what is the total number of Skoda cars sold in the year 2005?

Augmentation information:
 The table shows the sales figures for different Škoda models from 1991 to 2013. (Lack of column semantic information)

SQL:
 SELECT COUNT(*) FROM DF WHERE c_2005 != '-'

Answer: *3* ❌

Add augmentation information:
 1. c_2005: The sales figure for the year 2005

SQL:
 SELECT SUM(*) FROM DF WHERE c_2005 != '-'

SQL result: *492,111*

Final Answer: *492,111* ✅

Figure 9: Error case when the table augmentor fails to generate effective augmentation information

```

<table>
<caption>Oklahoma State Regents for Higher Education</caption>
<thead>
<tr><th>
Term_Expires</th><th>
Name</th><th>
Profession</th><th>
Hometown</th><th>
Office</th></tr>
</thead>
<tbody>
<tr><td>William Stuart Price</td><td>Businessman </td><td>Tulsa </td><td>2013
</td><td>
</td></tr>
<tr><td>Joseph L. Parker Jr.</td><td>Businessman </td><td>Tulsa </td><td>2014
</td><td>Chair </td></tr>
<tr><td>Bill W. Burgess Jr. </td><td>Attorney </td><td>Lawton </td><td>2011
</td><td>
</td></tr>
</tbody>
</table>

```

Figure 10: Demo for table format encoding used in ALTER, all tables are serialized in HTML format and include only K rows from the original table (where $K=3$ for the demo).

Algorithm 1 ALTER Workflow

Input: original table-question pair (T, Q) .

Output: predicted answer to the question \hat{A} .

```

1: function ALTER( $T, Q$ )
2:   #Function table organizer (Taborg) defined
3:   function TABORG( $T, Q$ )
4:     # Sample row index  $\hat{R}$  using embedding-based similarity
5:      $\hat{R} = \text{RowSample}(T, Q)$ 
6:     # Store table augmentation information in advance
7:      $Aug = \text{TabAug}(T_{\hat{R},:})$ 
8:      $(Aug_{c_1}, \dots, Aug_{c_{|C|}}, Aug_T) \leftarrow Aug$ 
9:      $\hat{C} = \text{ColFilter}(T_{\hat{R},:}, Q, Aug)$ 
10:     $sql = \text{CallLLM}(T_{\hat{R},\hat{C}}, Q, Aug)$ 
11:    return Execute( $sql$ )
12:  end function
13:  # Generate sub-queries with the query augmentor
14:   $\hat{R} = \text{RowSample}(T, Q)$ 
15:   $\{Q_i\}_{i=1}^m = \text{QueryAug}(T_{\hat{R},:}, Q)$ 
16:  # Run sub-queries in parallel
17:  for  $i$  in  $1, \dots, m$  do
18:     $T_i^{res} = \text{TabOrg}(T, Q_i)$ 
19:    # Get effective response for the sub-query
20:     $A_i^{res} = \text{CallLLM}(T_i^{res})$ 
21:  end for
22:  # Get accessible sub-table in the primary workflow
23:   $T^{res} = \text{TabOrg}(T, Q)$ 
24:  # Joint reasoner
25:   $\hat{A} = \text{CallLLM}(T^{res}, A_{1:m}^{res})$ 
26:  return  $\hat{A}$ 
27: end function

```

```

===== ** Step-back Augmentor ** =====
Below is a sub-table with rows randomly sampled from the original table. Based on
the sub-table, your task is to step back and paraphrase a question to a more
generic step-back question, which is easier to answer.

Sub-table: {Sub-table from in-context example}
Query: what is the next most populous district after haridwar?
New query: what districts are more populous than haridwar?

Sub-table: {Sub-table from in-context example}
Query: who was the only judge appointed by mckinley?
New query: which judge was appointed by mckinley?

Sub-table: {Sub-table from in-context example}
Query: was chuck bednarik or frank tripucka the first draft pick?
New query: who was the first draft pick?

Sub-table: {Sub-table from input}
Query:{Query}
New query:
===== ** Sub-query Augmentor ** =====
You are capable of converting complex queries into sub-queries. Below is a sub-
table with rows randomly sampled from the original table. Based on the sub-table,
decompose the original query into 2-3 complete sub-queries that can solve the
original query.

Sub-table: {Sub-table from in-context example}
Query: what was the time difference between the first place finisher and the
eighth place finisher?
New query: what was the time for the first place finisher?; what was the time for
the eighth place finisher?

Sub-table: {Sub-table from in-context example}
Query: other than william stuart price, which other businessman was born in tulsa
?
New query: where was william stuart price born in?; who was born in tulsa?

Sub-table: {Sub-table from in-context example}
Query: which canadian city had the most passengers traveling from manzanillo
international airport in 2013?
New query: how many passengers do each airline from canadian city have?; which
canadian city had the most passengers?

Sub-table: {Sub-table from input}
Query:{Query}
New query:

```

Figure 11: The prompt template for the query augmentor in WikiTQ

```

===== ** Schema Info ** =====
Instruction: Given the following table, you will add schema type about the
columns in the table.
Schema type includes:
- Numerical: consists of digits and numerical symbols like decimal points or
signs.
- Char: whether column content is a phrase or description.
- Date: whether column content represents time or date.

You need to output all the column names with metadata in angle brackets, e.g.
name<Char> launched<Date> count<Numerical>

Table: {Sub-table from input}
Output:
===== ** Semantic Info ** =====
Instruction: Given the following table, you need to first summarize the contents
of the table, then based on the summary, give a concluded description of each of
the columns.

The output should use the following format:
table summary: summary for table contents
column description: output all the column names with description in angle
brackets, e.g. launched<The launched date for the competition>

Table: {Sub-table from input}
Output:
===== ** Literal Info ** =====
Instruction: Below is a sub-table with rows sampled, you are required to infer
the data distribution and format from the sample data. Refine commonalities in
literal representations within each table column.

You need to output in the following format:
Column_name: Commonalities
e.g. championship: Names of golf tournaments are listed with some additional
information (e.g., 's open, classic)

Sub-table: {Sub-table from input}
Output:

```

Figure 12: The prompt template for three types of augmentation in the table augmentor in WikiTQ

```

===== ** Column Filter ** =====
Based on the Table below, your task is to accurately output columns related to
the query.
Approach this task as follows:
Read the query and extra information thoroughly and list every possible link from
query term to column in the Table.
Then based on the column linking, output all useful columns at last. Make sure
all columns in the linking step are included and every column is in the Table.

Table: {Sub-table from in-context example}
Extra information: The table contains information about the Hoot Kloot animated
series, including the episode number, title, director, and release year.
Column information:
1. Number: The episode number in the series
2. Title: The title of the episode
3. Directed_by_: The director of the episode
4. Released_: The release year of the episode

Query: what was the last title that sid marcus directed?
Column linking: the last title -> Released_, the last title-> Number, title ->
Title, sid marcus -> Directed_by_
Columns: Released_, Number, Title, Directed_by_

Table: {Sub-table from input}
Extra information: {Augmentation}

Query: {Query}
===== ** Augmented SQL Generation(Zero-shot) ** =====
Our ultimate goal is to answer the query based on the original table. Now we have
a sub-table with rows sampled from the original table, you are required to infer
the data distribution and format from the sample data of the sub-table. Based on
the augmentation information, carefully analyze the query and write an SQLITE3
SELECT SQL statement using table DF that completes the query. Directly output SQL
, do not add other string.

Sub-table: {Sub-table from column filter}
Augmentation information: {Augmentation}

Query: {Query}
SQL:

```

Figure 13: The prompt template for the column filter module and augmented SQL generation module inside the table organizer in WikiTQ

```

===== ** Reasoning from Query Augmentor ** =====
Below is a sub-table generated by executing the corresponding SQL. You need to
understand the logic behind the SQL filtering. Based on the sub-table, answer the
query using the final sub-table.

SQL Executed:
'''{SQL}'''
Sub-table: {Sub-table from executing SQL}
Query: {Sub-query}

Please provide a clear, complete statement in response to the query. If you
cannot answer the query based on the sub-table, return 'Cannot get answer from
sub-table'.
===== ** Joint Reasoner ** =====
Below is a sub-table generated by executing the corresponding SQL with extra
information may be useful. You need to understand the logic behind the SQL
filtering. Based on the sub-table and extra information provided, think step by
step and answer the query.

You should output in the following format:
Thought: your step by step thought
Answer: Only return the concise string instead of other format information. Do
not repeat the question.
Below is an example.

SQL Executed:
'''SELECT DISTINCT Type FROM DF WHERE Type != 'audio';'''
Sub-table: <table>
<thead>
<tr><th> Type </th></tr>
</thead>
<tbody>
<tr><td>video </td></tr>
<tr><td>audio/video </td></tr>
</tbody>
</table>
Extra information:
The payload types for audio include audio, video, and audio/video.

Query: other than audio, what type of payload types are there?
Thought: Based on the executed SQL query and the extra information provided, the
types include audio or video. Therefore, other than audio, the payload type is
video.
Answer: video

SQL Executed:
'''{SQL}'''
Sub-table: {Sub-table from executing SQL}
Extra information:
{Extra information from query augmentor}

Query: {Query}

```

Figure 14: The prompt template for joint reasoner in WikiTQ