TAP4LLM: Table Provider on Sampling, Augmenting, and Packing Semi-structured Data for Large Language Model Reasoning

Yuan Sui[1*†], **Jiaru Zou**[2*†], **Mengyu Zhou**[3‡], **Xinyi He**[4†],
Lun Du[5§], **Shi Han**[3], **Dongmei Zhang**[3]
[1] National University of Singapore [2] University of Illinois Urbana-Champaign
[3] Microsoft [4] Xi'an Jiaotong University [5] Ant Research
yuansui@comp.nus.edu.sg, jiaruz2@illinois.edu, hxyhxy@stu.xjtu.edu.cn
{mezho, shihan, dongmeiz}@microsoft.com, dulun.dl@antgroup.com

Abstract

Table reasoning tasks have shown remarkable progress with the development of large language models (LLMs), which involve interpreting and drawing conclusions from tabular data based on natural language (NL) questions. Existing solutions mainly tested on smaller tables face scalability issues and struggle with complex queries due to incomplete or dispersed data across different table sections. To alleviate these challenges, we propose **TAP4LLM** as a versatile pre-processor suite for leveraging LLMs in table-based tasks effectively. It covers several distinct components: (1) *table sampling* to decompose large tables into manageable sub-tables based on query semantics, (2) *table augmentation* to enhance tables with additional knowledge from external sources or models, and (3) *table packing & serialization* to convert tables into various formats suitable for LLMs' understanding. In each module, we design and compare several common methods under various usage scenarios, aiming to shed light on the best practices for leveraging LLMs for table-reasoning tasks. Our experiments show that our method improves LLMs' reasoning capabilities in various tabular tasks and enhances the interaction between LLMs and tabular data by employing effective pre-processing.

1 Introduction

The extensive and complex characteristics of the data are commonly represented in the format of structured data. **Table** is one of those fundamental and widely used semi-structured data types in different areas, such as relational databases, spreadsheet applications, and programming languages

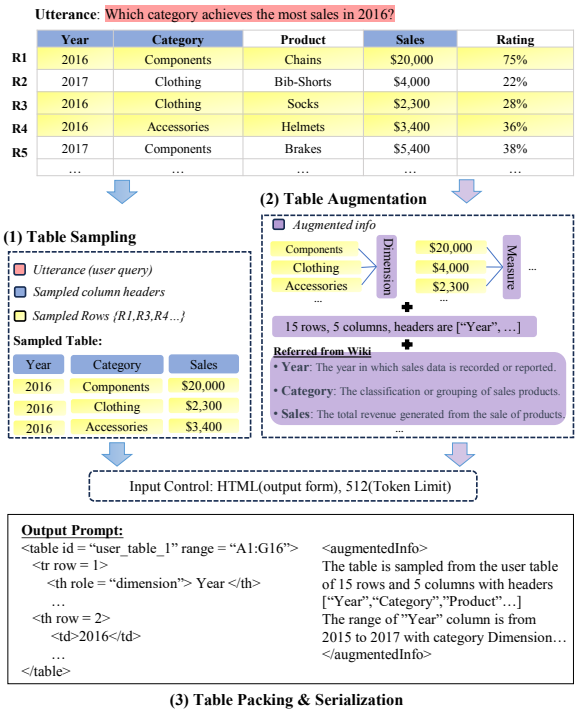

Figure 1: Demo of TAP4LLM Modules. (1) Table sampling: sample most relevant content. (2) Table augmentation: retrieve and add extra / meta information. (3) Table packing: serialize the sampled table and augment information into a string while controlling the number of tokens.

that handle data for various domains, including financial analysis (Zhang et al., 2020; Li et al., 2022), risk management (Babaev et al., 2019), healthcare analytics (Vamathevan et al., 2019), *etc*. Reasoning over tabular data has several important downstream tasks that are crucial to the field of natural language understanding (NLU) and information retrieval (IR), such as Table-based Question Answering (TQA) (Chen et al., 2020b; Iyyer et al., 2017; Ye et al., 2023a; Cheng et al., 2023), Table-based Fact Verification (TFV) (Chen et al., 2020a; Xie et al., 2022; Günther et al., 2021), Table-to-Text (Wang et al., 2021b), Text-to-SQL (Yu et al., 2018), Column Type & Relation Classification (Iida et al., 2021; Deng et al., 2020), *etc*.

Although there is remarkable progress on table

* Equal contribution.
† The contributions by Yuan Sui, Jiaru Zou, and Xinyi He have been conducted and completed during their internships at Microsoft.
‡ Corresponding author.
§ The contributions by Lun Du have been conducted and completed when he was a full-time researchers at Microsoft.

reasoning tasks with the development of large language models (LLMs) (Cheng et al., 2023; Ye et al., 2023a; Gemmell and Dalton, 2023), the existing solutions are mainly tested on small tables and cannot reflect the real challenges of table reasoning. For example, there are usually issues on scaling to large tables or handling complex queries that require gathering data from various parts of a table. How well do LLMs understand tables and how to leverage LLMs to work with table data remains an open question (Chen, 2022; Gong et al., 2020). Our research aims to explore this question and shed light on the best practices to leverage LLMs for table-reasoning tasks. Several specific practical issues are faced when leveraging LLMs for table reasoning tasks as follows.

First, which part of a table should be kept in the prompt? The full content of a table could be too lengthy and noisy to be included in the prompt. In addition, most LLMs have a limited input context window size in which an overlong table cannot fit. For large tables that satisfy the length constraint, it can still lead to unnecessary computations (of LLM on long prompt) and quality regressions (generation interfered from noisy input) when placing irrelevant table content (*w.r.t.* the task or query) in the prompt. To address the challenge, some sampling methods were proposed in ad-hoc ways. For example, truncating the input tables to contain only the first 20+ rows and 8 columns (Chen, 2022), or filtering relevant rows based on n-gram overlap between them and the utterance (Yin et al., 2020). To answer the question of which part to keep, we conduct a systematic study of different grounding and sampling algorithms in Section 2.1, and the experiments and findings can be found in Section 3.2.

Second, what additional/external knowledge could help LLMs better understand a table? The raw content of a table may contain ambiguous information (*e.g.*, abbreviations, domain-specific terms, column type, *etc*.) that requires further interpretation and clarification. As a result, direct reasoning with the raw table may lead to misinterpretation and hallucination by LLMs. To address this, some augmentation techniques were proposed to incorporate structured knowledge (Sui et al., 2023; Xie et al., 2022), common sense knowledge (Bian et al., 2023; Ogundare and Araya, 2023; Shen et al., 2023; Guo et al., 2023), and analytical knowledge (Jena et al., 2022; He et al., 2023) into training and inference processes. For example, (Jena et al., 2022) transforms existing tabular data to create diverse NL inference instances for better zero-shot performance. AnaMeta (He et al., 2023) infers implicit metadata behind raw table contents through field distribution and knowledge graph information. However, the techniques were proposed independently and there lack a comprehensive study that compares them and attempts to combine them to provide useful and diverse knowledge and thoughts for LLMs. We will discuss several augmentation methods in Section 2.2 and their corresponding experiments and findings can be found in Section 3.3.

Moreover, table augmentation plays a crucial role in avoiding LLMs to partitionally comprehend the semantics and distribution of the whole table solely based on table sampling (which may remove some essential rows/columns due to the limitations of the methods). It leverages the summarization, statistics, and metadata information derived from the whole table to represent high-level information, to compromise the trade-off with table sampling, which will intuitively decrease information entropy. We will discuss this essential trade-off in Section 3.5 and the experiments and findings can be found in Figure 3.

Third, how do we encode the table into a prompt? While sampling and grounding compress the table content, augmentation expands the prompt by adding more information. With a given token budget, one needs to find the balance to allocate available tokens between table content and augmented knowledge. Furthermore, the serialization format of the table also plays a critical role. It not only influences how well an LLM understands the table input (Sui et al., 2023), but also determines the string length of the serialized table and the augmented information. For example, as discussed in (Sui et al., 2023), table formats such as HTML (Aghajanyan et al., 2021) or XML are better understood by GPT models, but they also lead to increased token consumption. To pack a table into the prompt, these problems should be addressed with trade-offs (see Section 3.5).

To mitigate the aforementioned challenges, we propose **TAP4LLM** (**TA**ble **P**rovider for **L**arge **L**anguage **M**odels) as a versatile pre-processor suite for LLMs in table-based tasks. Specifically, there are three essential modules: (i) Table Sampling: decomposing large tables into manageable sub-tables based on query semantics; (ii) Table Augmentation: enhancing tables with additional knowledge from external sources or symbolic models; and (iii) Table Packing: convert ta-

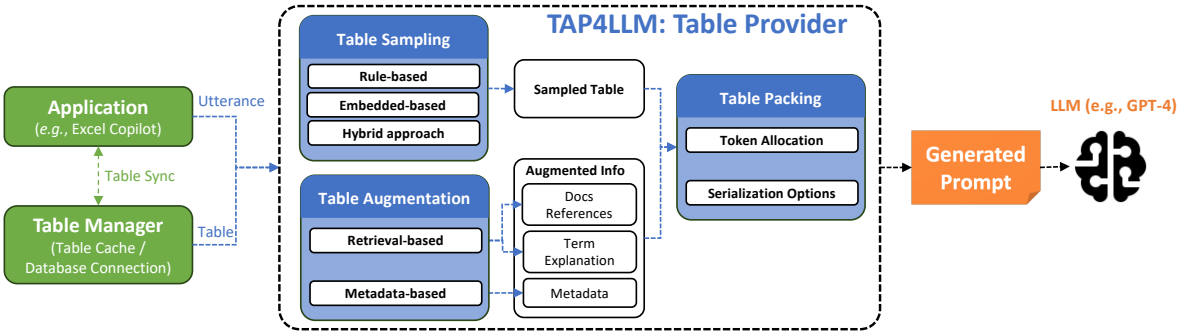

Figure 2: TAP4LLM Framework for Tabular Data. Note that "table sync" refers to the application (such as Excel Copilot) keeping its table data in sync with the table manager. The table manager acts as an intermediary, managing the data that is either stored locally in a cache or accessed through a database connection. This sync process is crucial for "interactive table reasoning" and for maintaining data integrity. The implications of this syncing process are further discussed in §F.2.

bles into various formats (*e.g.*, HTML, XML, Markdown, *etc*) suitable for LLMs' understanding while balancing the token allocation trade-off as well.

In each module, we design and compare new and existing methods for various scenarios across six distinct datasets. Through experiments in Section 3, we find that: (1) When using LLMs to process tables, it is more effective to concentrate on key rows and columns rather than overloading with extraneous data. For tasks that require high accuracy, semantic-based sampling typically enhances performance, whereas for tasks prioritizing low latency and minimal computational resources, rule-based sampling may be more suitable. (2) Integrating external knowledge of the tables can consistently improve the performance of table reasoning tasks by reducing hallucinations and factual inaccuracies in LLMs and improving general comprehension and analysis of tabular data. (3) A balanced distribution of tokens between table content and augmented information can help improve overall performance; We find a performance-optimized ratio (close to 5:5 or 4:6 between table content and augmentation tokens) often achieves the best performance across different settings.

In summary, our **main contributions** are:

- We propose TAP4LLM framework to improve the effectiveness of LLMs on tabular reasoning tasks by better sampling, augmentation and packing tables into input prompts.
- A comprehensive evaluation of each component is conducted. On average TAP4LLM achieves a 7.93% performance improvement.
- We summarize a table prompting guideline. For different real-world scenarios, we identify the corresponding optimal combination / setting of methods within each module.

2 TAP4LLM: Table Provider for LLMs

The overall architecture of TAP4LLM is defined as follows (as illustrated in Figure 2): Given a natural language query/utterance Q from applications (*e.g.*, Excel Copilot) and a table T from Table Manager (*e.g.*, table cache or database connection), our system incorporates three core components as follows:

- **Table Sampling**: Decompose a large table T into a sub-table T' with specific rows and columns.
- **Table Augmentation**: Incorporate relevant external knowledge, metadata, and attributes about the original table T explicitly.
- **Table Packing**: Convert tables into various formats suitable for LLMs' understanding while control the token allocation for table sampling and augmentation.

2.1 Table Sampling

In table sampling, a subset of top-ranked rows and

the ranked list to form the sub-table $T_{r,c}$. Specifically, we classify multiple variants for table sampling as three following categories:

2.1.1 Rule-based Sampling

Rule-based sampling refers to table sampling based on predefined criteria or rules. These methods follow the established patterns or criteria for data selection. Specifically, we consider three common rule-based sampling methods incoporated into our table sampling module: (1) *Random Sampling*, (2) *Evenly Sampling*, and (3) *Content Snapshot & Synthetically Sampling*. The detailed description can be found in Appendix A.

2.1.2 Embedding-based Sampling

Instead of adhering to strict rules or criteria in rule-based sampling, embedding-based methods leverage the semantic and contextual representation of each row and column. Specifically, let T be a table where R_T is the set of rows and C_T is the set of columns. Let $E : R_T \cup C_T \rightarrow \mathbb{R}^d$ be an embedding function that maps each row or column to a d-dimensional vector by capturing its semantic content. By mapping each row or column to vectors, this method harnesses the power of spatial relationships within the embedding space to guide sampling decisions.

Here, we propose three variant methods as shown below: (1) *Semantic-based Sampling*: Semantic-based Sampling is a tailored approach emphasizing the semantics relevance of row/-columns to the utterance. The process is exactly illustrated in Eq. 1. Note that the default query-based sampling is the row-based method. In our experiments, we also study the column grounding shown in Table 1. (2) *Centroid-based Sampling*: The goal of centroid-based sampling is to ensure the preservation of data diversity. We use *K-Means* (MacQueen et al., 1967) to partition the set of embeddings into n clusters C_n. For each cluster C_i, we select the top-K rows or columns based on the closeness to the centroid. (3) *Hybrid-approach*: The Hybrid approach marries the specificity of semantic-based sampling with the broad representations of centroid-based sampling. Specifically, the top-K rows or columns are selected based on a combination metric $h(r, c, u)$ measuring the directional distance to cluster centroid c and the semantic similarity to the utterance u. We formulate the measuring metric as:

$$h(r, c, u) = \alpha(\frac{1}{1 + D(r, c)}) + \beta S(r, u) \quad (2)$$

where $D(r, c)$ measures the directional distance (*e.g.*, Euclidean distance) between selected rows or columns and cluster centroid in embedding space, and $S(r, u)$ measures the semantic similarity between rows/columns and the utterance. The weights α and β provide flexibility in prioritizing between contextual relevance and diversity. Without further specification, we set $\alpha = 0.3$ and $\beta = 0.7$ in our experiments.

2.1.3 LLM-based Sampling

LLMs have proven effective in tabular reasoning (Ye et al., 2023a), utilizing their capabilities to predict row and column indices for efficient sub-table extraction. However, relying on LLMs for pre-processing significantly increases computational costs. Additionally, using LLMs to predict indices introduces challenges such as token limitations, noisy information, and the need for further table pre-processing, turning the task into a recursive loop. Despite this method not being ideally suited to our scope, we still consider it a strong baseline, albeit at the expense of time.

2.2 Table Augmentation

Table augmentation enhances LLM reasoning by adding extra knowledge to the input table and query. In the table augmentation module, we categorize different knowledge aspects into three main categories: Metadata-based, Retrieval-based, and Self-consistency-based (See Table 7 in Appendix). We will describe each category in detail below.

2.2.1 Metadata-based Augmentation

Tabular data analysis relies on accurately understanding field semantics and identifying common patterns in everyday analysis (He et al., 2023). Following AnaMeta (He et al., 2023) using a range of knowledge-fusion language models for metadata inference, we consider five main metadata-based augmentation types and leverage LLMs for zero-shot inference using metadata instruction as clues: *Dimension / Measure*, *Semantic Field Type*, *Table Size*, *Statistics Feature*, *Header Hierarchy*. The detailed description for each meta-data augmentation type can be found in Appendix C.

Table 1: Comparative results of the table sampling methods. The term "w/ Column Grounding" refers to the method consider both row-based and column-based sampling (sometimes referred to as "grounding"). "GPT-3.5" refers to the OpenAI released model gpt-3.5-turbo-32k, with 32k token-sized context window; In contrast, "GPT-3.5 truncated" refers to gpt-3.5-turbo, with 4k token-sized context window, where most tables will be truncated according to this token limitation. The top-3 performances on each dataset are highlighted in green, with the best performance being both bold and underlined.

Sampling Type	Table Sampling Methods	SQA	FEVEROUS	TabFact	HybridQA	ToTTo
Rule-based Sampling	Random Sampling	27.30%	60.30%	55.17%	23.60%	40.12%
	Evenly Sampling	26.72%	61.87%	54.63%	5.32%	29.41%
	Content Snapshot (Yin et al., 2020)	28.24%	63.10%	56.92%	23.40%	47.51%
Embedding-based Sampling	Centroid-based Sampling	28.10%	63.50%	55.40%	24.03%	48.30%
	Semantic-based Sampling w/ Column Grounding	28.32%	63.32%	59.80%	24.32%	49.14%
		29.12%	64.74%	60.23%	**25.14%**	**53.42%**
	Hybrid Sampling	28.79%	**65.34%**	**61.37%**	24.71%	51.63%
LLM-based Sampling	LLM-Decomposer (Ye et al., 2023b)	27.98%	62.34%	58.74%	24.98%	48.13%
-	No sampling (GPT-3.5)	27.60%	60.12%	56.20%	14.10%	47.42%
	No sampling (GPT-3.5, truncated)	23.54%	43.54%	52.12%	23.12%	30.42%

2.2.2 Retrieval-based Augmentation

Large Language Models have occasionally been observed to generate hallucinated or factually inaccurate text (Zhou et al., 2021; Zhao et al., 2023). To mitigate these issues, several works have proposed to strengthen LLMs with information retrieval systems (Shi et al., 2023; Jiang et al., 2023b; Nakano et al., 2022), which enables LLMs to retrieve relevant content from an external repository (knowledge corpus). It has been verified that retrieval-augmented LLMs can generate texts in response to user input with fewer hallucinations (Nakano et al., 2022). Furthermore, by incorporating customized private data resources, retrieval-augmented LLMs can respond to in-domain queries that cannot be answered by LLMs trained with public data. As previous works (Nakano et al., 2022; Shi et al., 2023; Jiang et al., 2023b) suggested, LLMs can generate more factual answers by feeding the references retrieved from the external corpus.

In TAP4LLM, we have fortified the document retrieval capabilities of LLMs and consider two components: (1) *document references*: provide supplemental relevant web pages as the references for the given table; (2) *term explanation*: explain strange/ambiguous term in the given table. We utilize technologies including vector databases (Wang et al., 2021a) and LangChain (LangChain, 2022) to facilitate the retrieval of pertinent information from Wikipedia[1]. The details for document references and term explanation can be found in Appendix B.

[1] https://www.wikipedia.org/

2.2.3 Self-consistency-based Augmentation

We follow (Sui et al., 2023) to implement the self-consistency-based augmentation approach. Specifically, we append the instruction *"Identify critical values and ranges of the last table related to the statement"* to the initial prompt and forward it to the LLM. The output generated from this instruction is then incorporated back into the prompt. Following this, we then re-forward the enriched prompt to LLMs — containing both the initial query and the newly generated insights, along with task-specific instructions for further processing.

2.3 Table Packing

The table packing module is motivated by the need to preserve efficient reasoning without altering the LLM architecture. First, we apply this module to manage token-limit allocation for the table sampling and augmentation. To achieve this, we conduct an empirical study to determine the optimal ratio between the sub-table length and the length of the augmentation information, as illustrated in Figure 3. The packing process is regulated by a user-defined token limit parameter, which sets the maximum truncated token length. Additionally, we are inspired by the study (Sui et al., 2023), which highlights that using markup languages such as *HTML* or *XML* significantly improves generation quality in comparison to TQA and TFV. In line with this, TAP4LLM supports multiple serialization formats, including HTML, XML, JSON, CSV, NL+Sep (a common option, *e.g.*, using '|' as a cell/column separator), and Markdown, *etc*.

Table 2: Comparative results of the table augmentation methods. We use a semantic-based sampling method without augmentation as the baseline. The term "Delta" refers to the performance gap between each method and the baseline. The top-3 performance gaps on each dataset are highlighted in green, with the best performance being underlined. Note that since only the ToTTo dataset contains hierarchical headers, we only provide the "header hierarchy" method on this dataset. "D/M + SF" refers to Dimension/Measure+ Semantic Field Type.

Augmentation Aspect	SQA		FEVEROUS		TabFact		HybridQA		ToTTo	
	Acc	Delta	Acc	Delta	Acc	Delta	Acc	Delta	BLEU-4	Delta
baseline	28.32%	0.00%	63.32%	0.00%	59.80%	0.00%	24.32%	0.00%	49.14%	0.00%
D/M + SF	30.12%	1.80%	65.72%	2.40%	**62.67%**	2.87%	26.12%	1.80%	51.25%	2.11%
Table Size	28.85%	0.53%	63.40%	0.08%	60.30%	0.50%	24.94%	0.62%	49.03%	-0.11%
Statistics Feature	31.22%	2.90%	**66.51%**	3.19%	62.33%	2.53%	26.13%	1.81%	50.57%	1.43%
Header Hierarchy	-	-	-	-	-	-	-	-	48.64%	-0.50%
Docs References	**33.45%**	5.13%	63.13%	-0.19%	61.32%	1.52%	25.12%	0.80%	52.74%	3.60%
Term Explanations										
- LLM-based	31.59%	3.27%	64.12%	0.80%	62.32%	2.52%	26.24%	1.92%	**53.21%**	4.07%
- Heuristics-based	29.59%	1.27%	63.72%	0.40%	61.58%	1.78%	25.24%	0.92%	51.21%	2.07%
Self Prompting	30.45%	2.13%	65.24%	1.92%	62.32%	2.52%	**26.64%**	2.32%	52.36%	3.22%

3 Experiments

In this section, we first present the experimental setup, followed by an extensive comparison between baselines within each module in TAP4LLM. Additionally, we conduct an ablation study and provide a thorough evaluation of the performance of TAP4LLM. For further details on the experimental setup and additional experiments, please refer to Appendix D and E.

3.1 Experiment Settings

Datasets. We evaluate TAP4LLM on five TQA & TFV datasets: Sequential Question Answering (SQA) (Iyyer et al., 2017), HybridQA (Chen et al., 2020b), TabFact (Chen et al., 2020a), ToTTo (Parikh et al., 2020). Additionally, we set up TAP4LLM on a Text-to-SQL database Spider (Yu et al., 2018), detailed in E.4. The statistics of the datasets are given in Table 6, and the details of the datasets and metrics are described in Appendix D.1.
Models. We select state-of-the-art LLMs that have been widely studied in text generation and reasoning, including multiple GPT-series models and advanced open-source LLMs. Details of the tested models and embedding methods are provided in Appendix D.2, while the experimental results for open-source LLMs are available in Appendix E.3.

3.2 Results of Table Sampling

As shown in Table 1, we perform comparative experiments on various table sampling methods, leading to the following key observations: (1) *Semantic-based sampling* with column grounding outperforms other sampling methods across all datasets by effectively selecting table parts most relevant to queries. *Centroid-based sampling* also shows competitive results by clustering data points within tables, though it lacks query-table relevance consideration. Moreover, when combining these two strong variants (*hybrid sampling*), it shows the most powerful capability. (2) The *rule-based sampling* method *content snapshot*, while not as precise in capturing query-specific information, offers a promising, efficient alternative by focusing on essential table content through n-gram overlap, without the need for complex embedding calculations. (3) Direct encoding methods, including using GPT-3.5-turbo with a 32k token limit or a 4k token-sized context window with truncation, demonstrate inferior performance. This suggests that while they can encompass more table information, they may introduce noise or lose critical context, undermining the table reasoning process and highlighting the importance of strategic sampling for optimal LLM performance.

3.3 Results of Table Augmentation

For the comparative experiments of table augmentation methods, we use the semantic-based sampling method as the baseline and report the performance gap between adding each augmentation method or not. Table 2 provides several key insights, summarized as follows: (1) Table augmentation methods further improve LLM's reasoning ability after sampling. For example, *D/M + SF* achieves higher accuracy across all six datasets (a most significant increase on TabFact +2.87%). *Docs References* and *Term Explanations* add mean-

ingful context and semantic understanding to the model's processing of tables, with (SQA +5.13%, ToTTo +4.07%). The *Self-Prompting* further exemplifies the potential for iterative improvement in query and response generation. However, not all augmentation methods yield positive outcomes. *Table Size* offers minimal performance enhancement and *Header Hierarchy* shows that introducing a hierarchy may complicate the model's ability to process the tabular information in some contexts, possibly by adding unnecessary complexity. (2) Additionally, the comparison of cell selection methods for *Term Explanations* highlights the superior performance of LLM-based selection over heuristic approaches. We find that LLM-based cell selection outperforms the heuristics-based cell selection with improvements in "Delta" ranging from 0.80% to 4.07%. While achieving higher performance, the LLM-based method also increases the calling budget as it requires additional LLM calls. These results indicate that the method's effectiveness varies with the dataset. i.e. It's beneficial for datasets requiring complex text understanding and generation (SQA and ToTTo). However, its impact is less distinct or even slightly negative in datasets involving different types of data or nuanced tasks (FEVEROUS and HybridQA).

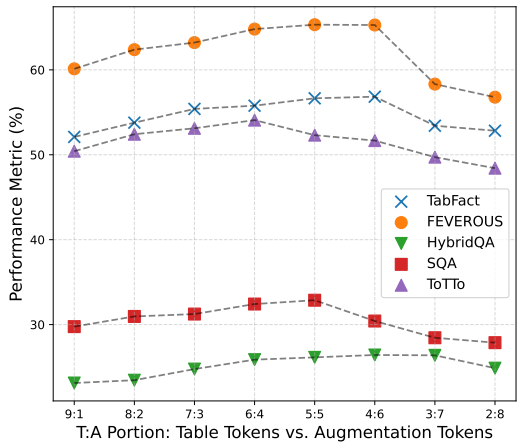

Figure 3: Token Allocation. T:A refers to the ratio of upper #token limitations of sampled table *vs.* augment info.

Through the experiment results, we also observe that different augmentation methods perform well on the same dataset. For example, *D/M + SF*, *Statistics Feature*, *Term Explanation* and *Self Prompting* all show significant improvement on the TabFact dataset. This suggests that combining multiple augmentation methods may have cumulative effects, leading to improved performance. In this work, we only report one simple intuitive approach - appending all augmentation information together into the prompt. We leave more fine-grained combinations for future exploration.

3.4 Ablation Study of TAP4LLM

As shown in Table 3, we conduct an ablation study to evaluate the impact of various components on the performance of TAP4LLM. Each row represents the model's performance with the removal of a specific component. Our findings indicate that all components contribute to the overall effectiveness of the model, with certain components, such as table sampling and table augmentation, being particularly critical. The study also reveals that each dataset responds differently to the removal of features, underscoring the importance of a tailored design when optimizing for specific datasets. We also report the performance using the most optimal combination of table sampling and augmentation for each dataset, as presented in Table 3.

3.5 Trade-offs between Token Allocation

We employ five table datasets to explore the trade-off between token allocation for table sampling and table augmentation, as illustrated in Figure 3. We observe that: (1) A balanced token distribution between the table and augmentation (approximately 5:5 or 4:6, referred to as the *balanced T:A ratio*) generally achieves the best performance across all five datasets. This suggests that carefully managing token allocation can enhance LLM performance. (2) Diminishing returns are observed when an excessive number of tokens are allocated to augmentation information (e.g., a 3:7 ratio), leading to a decline in performance. This indicates that beyond a certain point, additional augmentation tokens may no longer be beneficial and could detract from the core table content.

The trade-off we analyze above reflects a broader principle in data processing and machine learning: the balance between information overload and information scarcity. Over-augmentation can introduce noise, making it harder to identify key patterns or insights, while excessive sampling may lead to an incomplete or biased understanding of the data. It is important to note that the optimal T:A ratio may vary across datasets, as each has unique characteristics that make certain ratios more effective.

3.6 Computational Implications

Table 4 demonstrates the bottleneck of computation for TAP4LLM. Specifically, The first row calculates the average LLM calls conducted during

Table 3: Ablation results on five table datasets using gpt-3.5-turbo model. Similar to Table 2, the lowest accuracy on each dataset is bold. The top-3 decreasing gap (delta) on each dataset are highlighted in red, with the lowest performance being underlined. The performance of golden combination of table sampling and augmentation ("hybrid-sampling + all-augmentation") is reported in the first row.

Components of TAP4LLM	SQA		FEVEROUS		TabFact		HybridQA		ToTTo	
	Acc	Delta	Acc	Delta	Acc	Delta	Acc	Delta	BLEU-4	Delta
All	34.12%	0.00%	68.32%	0.00%	64.78%	0.00%	27.87%	0.00%	54.93%	0.00%
w/o table sampling	**26.54%**	-7.58%	**61.54%**	-6.78%	**58.12%**	-6.66%	**24.12%**	-3.75%	**48.47%**	-6.46%
w/o table augmentation - all	29.12%	-5.00%	63.74%	-4.58%	60.23%	-4.55%	25.14%	-2.73%	53.42%	-1.51%
w/o table augmentation - metadata-based	33.87%	-0.25%	64.38%	-3.94%	62.78%	-2.00%	26.98%	-0.89%	53.42%	-1.51%
w/o table augmentation - retrieval-based	31.42%	-2.7%	66.23%	-2.09%	62.97%	-1.81%	26.33%	-1.54%	52.67%	-2.26%
w/o table packing	31.87%	-2.25%	67.42%	-0.90%	63.28%	-1.50%	26.32%	-1.55%	52.87%	-2.06%

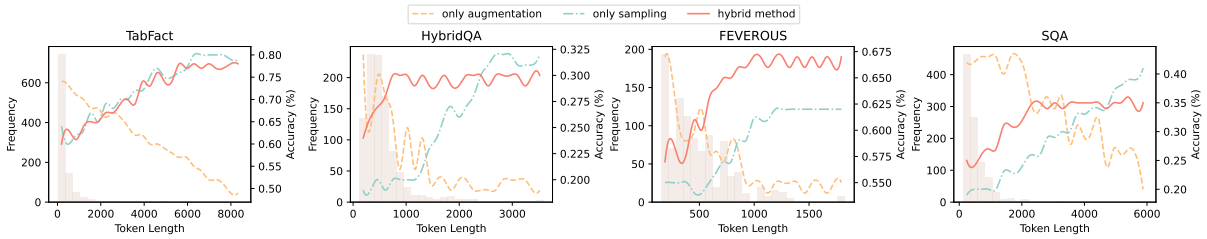

Figure 4: Comparative Analysis of Model Performance Across TabFact, HybridQA, FEVEROUS and SQA. The series of graphs illustrates the frequency distribution of token lengths alongside the LLM performance (%) for three distinct methods: only sampling, only augmentation, and the hybrid method. Each subplot corresponds to a different dataset, depicting how table token length impacts model accuracy for various data augmentation and sampling strategies. Note that the "only augmentation" method refers to adding only the augmentation information to the prompt, without using any sampling method.

the Table Augmentation with Docs References and Term Explanations, which contributes to $N + C$ times LLM calling. Here, N is the number of terms detected by the LLM, and C is a constant equal to 2 if there is no error occurs when calling the API. We also measure the token usage for each LLM call during the Table Sampling. Since the Table Sampling module only retrieves the top-k-related rows/columns to reconstruct a sub-table for LLM calling, this ensures token expenditure without compromising the model's performance, especially in cases where thousands of rows exist in a table. The second row of the table 4 shows the average token usage per question needed by TAP4LLM after table sampling.

Table 4: Efficiency of multiple preprocessing steps in TAP4LLM

Dataset	SQA	FEVEROUS	TabFact	HybridQA
Average LLM Calls	4.7	5.2	5.3	8.4
Average Token Usage	637	512	417	742

3.7 Large Table Analysis in TAP4LLM

Compared to smaller tables, large tables can grow to immense sizes, making them more difficult to maintain and reason over tabular data. In designing TAP4LLM, performance optimization in this context is essential. Figure 4 presents the distribution of token counts from the table across the five datasets, while also illustrating the impact of token numbers on LLM performance in three distinct settings. We can observe that: (1) Shorter token lengths dominate the datasets, indicating the prevalence of text entries is relatively brief. (2) Augmentation techniques excel with these shorter lengths by providing focused, enriched contexts that facilitate better model learning from simpler inputs. In contrast, sampling methods prove more effective for larger tables, suggesting that they help manage data complexity by focusing on relevant data segments. (2) The hybrid method shows consistent performance across various token lengths, highlighting its ability to leverage the strengths of both augmentation and sampling for robust performance enhancement.

4 Related Work

Large Language Models for Tabular Data Following the line of LLMs in natural language processing, researchers have also explored large models for various modalities like vision (Gong et al., 2023; Kirillov et al., 2023) and speech (Huang et al., 2023). From a technical standpoint, their ability to generate human-like text has opened new

vistas of possibilities for processing tabular data. Nevertheless, it is non-trivial to directly employ the vanilla LLMs in the tabular area for two reasons: (i)-Global Table Understanding: the GPTs are known to suffer from the limited token length and thus, they can not read a whole large table, making them hard to understand the global tabular information. (ii)-Generalized to Tabular Domain: Second, their training processes are tailored for natural languages and thus, they are less generalizable when handling tabular data. There have been several works (Hu et al., 2023; Zhong et al., 2017; Li et al., 2023b,a) developed to integrate natural language for tabular data analysis.

Table Augmentation Table augmentation is a technique used to improve the generalization performance and robustness of machine learning models. To enhance the performance and capabilities of LLMs in various domains, various explorations have been done to augment their knowledge grounding. It involves incorporating structured knowledge (Sui et al., 2023; Xie et al., 2022), commonsense knowledge (Bian et al., 2023; Ogundare and Araya, 2023; Shen et al., 2023; Guo et al., 2023), and analytical knowledge (He et al., 2023; Jena et al., 2022) into the pre-training and inference processes. For example, (Jena et al., 2022) proposes to semi-automatically transform existing tabular data to create diverse/inventive natural language inference instances for better zero-shot performance. (He et al., 2023) proposes a multi-tasking Metadata model that leverages field distribution and knowledge graph information to accurately infer analysis metadata for tables, and then demonstrates its deployment in a data analysis product for intelligent features.

5 Conclusion

In this paper, we propose TAP4LLM (Table Provider for LLM) as a powerful toolkit designed to enhance the interaction between LLMs and structured table data. It provides optimized prompt designs and robust functionalities to ensure high-quality outputs when LLMs process table-related inputs. We believe that TAP4LLM has the potential to significantly improve table modeling and exploratory data analysis (EDA), with applications in various domains.

Limitations

Code generation-based methods (Cheng et al., 2023; Gemmell and Dalton, 2023; He et al., 2024) have been proposed to leverage LLMs to convert natural language queries into executable code or structured representations. We believe that semantic parsing or code generation is an important research direction. However, due to the page limits, we will leave this topic to further exploration. Additionally, our empirical study is mostly designed for English, rather than multilingual scenarios. The conversation on multilingual capabilities will also be part of future exploration.

Ethics Statement

All the datasets used in this paper are public and have been reviewed to ensure they do not contain any personally identifiable information or offensive content. However, as these datasets are sourced from the Internet, potential bias may still be present. Furthermore, despite our careful review, the process of table augmentation with LLMs throughout may inadvertently introduce inappropriate information into the preprocessed data. In addition, all the experiments in this paper are run on GPU clusters with 8 NVIDIA A100 GPUs.

References

New and improved embedding model. https://openai.com/blog/new-and-improved-embedding-model.

Armen Aghajanyan, Dmytro Okhonko, Mike Lewis, Mandar Joshi, Hu Xu, Gargi Ghosh, and Luke Zettlemoyer. 2021. Htlm: Hyper-text pre-training and prompting of language models.

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.

Dmitrii Babaev, Maxim Savchenko, Alexander Tuzhilin, and Dmitrii Umerenkov. 2019. E.T.-RNN: Applying Deep Learning to Credit Loan Applications. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 2183–2190, New York, NY, USA. Association for Computing Machinery.

Ning Bian, Xianpei Han, Le Sun, Hongyu Lin, Yaojie Lu, and Ben He. 2023. ChatGPT is a Knowledgeable but Inexperienced Solver: An Investigation of Commonsense Problem in Large Language Models. ArXiv:2303.16421 [cs].

Wenhu Chen. 2022. Large language models are few(1)-shot table reasoners.

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.

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.

Zhihong Chen, Feng Jiang, Junying Chen, Tiannan Wang, Fei Yu, Guiming Chen, Hongbo Zhang, Juhao Liang, Chen Zhang, Zhiyi Zhang, et al. 2023. Phoenix: Democratizing chatgpt across languages. *arXiv preprint arXiv:2304.10453*.

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. ArXiv:2210.02875 [cs].

Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. Turl: Table understanding through representation learning.

Rui Ding, Shi Han, Yong Xu, Haidong Zhang, and Dongmei Zhang. 2019. QuickInsights: Quick and Automatic Discovery of Insights from Multi-Dimensional Data. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, pages 317–332, New York, NY, USA. Association for Computing Machinery.

Carlos Gemmell and Jeffrey Dalton. 2023. Generate, Transform, Answer: Question Specific Tool Synthesis for Tabular Data.

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.

Tao Gong, Chengqi Lyu, Shilong Zhang, Yudong Wang, Miao Zheng, Qian Zhao, Kuikun Liu, Wenwei Zhang, Ping Luo, and Kai Chen. 2023. Multimodal-gpt: A vision and language model for dialogue with humans. *arXiv preprint arXiv:2305.04790*.

Michael Günther, Maik Thiele, Julius Gonsior, and Wolfgang Lehner. 2021. Pre-trained web table embeddings for table discovery. In *Fourth Workshop in Exploiting AI Techniques for Data Management*, pages 24–31.

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. 2023. How Close is ChatGPT to Human Experts? Comparison Corpus, Evaluation, and Detection. 4 citations (Semantic Scholar/arXiv) [2023-02-20] 4 citations (Semantic Scholar/DOI) [2023-02-20] arXiv:2301.07597 [cs].

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.

Xinyi He, Jiaru Zou, Yun Lin, Mengyu Zhou, Shi Han, Zejian Yuan, and Dongmei Zhang. 2024. Conline: Complex code generation and refinement with online searching and correctness testing. *arXiv preprint arXiv:2403.13583*.

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

Jamie Hoelscher and Amanda Mortimer. 2018. Using Tableau to visualize data and drive decision-making. *Journal of Accounting Education*, 44:49–59.

Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. 2023. Chatdb: Augmenting llms with databases as their symbolic memory. *arXiv preprint arXiv:2306.03901*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Rongjie Huang, Mingze Li, Dongchao Yang, Jiatong Shi, Xuankai Chang, Zhenhui Ye, Yuning Wu, Zhiqing Hong, Jiawei Huang, Jinglin Liu, et al. 2023. Audiogpt: Understanding and generating speech, music, sound, and talking head. *arXiv preprint arXiv:2304.12995*.

IDEA-CCNL. 2023. Fengshenbang-lm. `https://github.com/IDEA-CCNL/Fengshenbang-LM`.

Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. Tabbie: Pretrained representations of tabular data.

Baichuan Intelligence. 2023. Baichuan-7b. `https://github.com/baichuan-inc/baichuan-7B`.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based Neural Structured Learning for Sequential Question Answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 1821–1831, Vancouver, Canada. Association for Computational Linguistics.

Aashna Jena, Vivek Gupta, Manish Shrivastava, and Julian Martin Eisenschlos. 2022. Leveraging Data Recasting to Enhance Tabular Reasoning.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023a. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. Active retrieval augmented generation.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. 2023. Segment anything. *arXiv preprint arXiv:2304.02643*.

LangChain. 2022. Langchain. `https://blog.langchain.dev/`.

Hongxin Li, Jingran Su, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. 2023a. Sheetcopilot: Bringing software productivity to the next level through large language models. *arXiv preprint arXiv:2305.19308*.

Jinyang Li, Binyuan Hui, Reynold Cheng, Bowen Qin, Chenhao Ma, Nan Huo, Fei Huang, Wenyu Du, Luo Si, and Yongbin Li. 2023b. Graphix-t5: Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing. *arXiv preprint arXiv:2301.07507*.

Liyao Li, Haobo Wang, Liangyu Zha, Qingyi Huang, Sai Wu, Gang Chen, and Junbo Zhao. 2022. Learning a Data-Driven Policy Network for Pre-Training Automated Feature Engineering. In *The Eleventh International Conference on Learning Representations*.

James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. Webgpt: Browser-assisted question-answering with human feedback.

Oluwatosin Ogundare and Gustavo Quiros Araya. 2023. Comparative Analysis of CHATGPT and the evolution of language models.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Ankur P. Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. Totto: A controlled table-to-text generation dataset.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Xinyue Shen, Zeyuan Chen, Michael Backes, and Yang Zhang. 2023. In ChatGPT We Trust? Measuring and Characterizing the Reliability of ChatGPT. ArXiv:2304.08979 [cs].

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. REPLUG: retrieval-augmented black-box language models. *CoRR*, abs/2301.12652.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2023. Evaluating and enhancing structural understanding capabilities of large language models on tables via input designs.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Jessica Vamathevan, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran, George Lee, Bin Li, Anant Madabhushi, Parantu Shah, Michaela Spitzer, and Shanrong Zhao. 2019. Applications of machine learning in drug discovery and development. *Nature Reviews Drug Discovery*, 18(6):463–477. Number: 6 Publisher: Nature Publishing Group.

Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, et al. 2021a. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2614–2627.

Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021b. TUTA: Tree-based Transformers for Generally Structured Table Pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1780–1790. ArXiv:2010.12537 [cs].

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023a. Large Language Models are Versatile Decomposers: Decompose Evidence and Questions for Table-based Reasoning.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023b. Large Language Models are Versatile Decomposers: Decompose Evidence and Questions for Table-based Reasoning. ArXiv:2301.13808 [cs].

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online. 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. *arXiv preprint arXiv:1809.08887*.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. In *The Eleventh International Conference on Learning Representations*.

Tianping Zhang, Yuanqi Li, Yifei Jin, and Jian Li. 2020. AutoAlpha: an Efficient Hierarchical Evolutionary Algorithm for Mining Alpha Factors in Quantitative Investment. ArXiv:2002.08245 [q-fin].

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models.

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

Chunting Zhou, Graham Neubig, Jiatao Gu, Mona Diab, Francisco Guzmán, Luke Zettlemoyer, and Marjan Ghazvininejad. 2021. Detecting hallucinated content in conditional neural sequence generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1393–1404, Online. Association for Computational Linguistics.

Jiaru Zou, Mengyu Zhou, Tao Li, Shi Han, and Dongmei Zhang. 2024. Promptintern: Saving inference costs by internalizing recurrent prompt during large language model fine-tuning. *arXiv preprint arXiv:2407.02211*.

A Rule-based Sampling

Rule-based sampling refers to table sampling based on predefined criteria or rules. These methods follow the established patterns or criteria for data selection. We consider three common rule-based sampling methods as follows: (1) *Random Sampling*, by selecting rows from a table, with each having an equal probability of being selected. To increase the quality of this baseline, we repeat the random selection for a user-specified amount of time and return the sub-table with the highest combined score among all the randomly computed subtables. (2) *Evenly Sampling*: It samples rows from a table by alternating between the top (r_1) and bottom rows (r_n) and moving towards the middle until reaching a set token limit. Compared to random sampling, it helps to balance the proportions of each field in the dimension column of the table (*i.e.* rows are selected at regular intervals), ensuring a uniform distribution of the sample accross the entire table. (3) *Content Snapshot & Synthetically Sampling*: Content snapshot (Yin et al., 2020) is a text-matching based method for retrieving subtables. For our empirical analysis, we construct the content snapshot K rows based on their relevance to the utterance using n-gram overlap ratio. Specifically, for $K > 1$, top-K rows with the highest n-gram overlap ratio are selected. For $K = 1$, a synthetic row is composed by selecting the cell values from each column with the highest n-gram overlap with the utterance. The comparative results can be found in Table 1.

B Retrieval-based Augmentation

B.1 Docs References

This process involves associating tables with relevant documents or sources for in-depth insights or references. For example, suppose we have a table titled "2023 Fortune 500 Companies". This table contains various information about the top 500

companies as ranked by Fortune in 2023, including their revenue, number of employees, and market capitalization. Docs references could fetch the actual 2023 Fortune 500 list from the Fortune website, Wikipedia pages discussing the Fortune 500 concept and its criteria, or analytical articles discussing the companies on the 2023 list. In our setting, we leverage Langchain (LangChain, 2022) to retrieve wiki pages from wikipedia.org. We craft queries by concatenating the table header and the table's title into a single string. These queries are then used to identify and fetch the relevant Wikipedia pages, which act as informative document references in our study.

B.2 Term Explanation

Compared to the docs references, term explanation focuses on providing definitions and explanations for specific strange terms or values in the table cells. For example, if a cell mentions a technical term or an acronym, the term explanation module could source a brief definition or background from reliable web sources (such as Wikipedia, wolfram,*etc*) on that term, ensuring that the strange term will not be forwarded to LLMs. To ensure the efficacy and accuracy of term explanations, we introduce two distinct approaches for selecting the cell that is required to be explained, *LLM-based Cell Selection* and *Heuristics-based Cell Selection*. The comparative experiment results of these two variants can be found in Table 2.

1) LLM-based Cell Selection Module: To pinpoint the exact cell warranting explanation, we harness the capabilities of LLMs. The selection prompt is meticulously constructed, taking into account various factors including: (1) Cell Position; (2) Cell Content; (3) Cell Formatting; (4) Cell Context; (5) Cell Properties. A detailed description and the specific prompt utilized to determine which cells require explanation can be found in Table 5.

2) Heuristics-based Cell Selection: Inspired by the methodology presented in (Herzig et al., 2020), we introduce a heuristics-based cell selection, which is predicated upon the following criteria: (1) Explicit Mention: whether the cell's value is explicitly referenced in the query. (2) Comparative Value: whether the cell's value is greater or less than a value mentioned in the query. (3) Superlative Value: whether the cell's value represents a maximum or minimum across the entire column, especially when the query incorporates superlative terms.

Table 5: LLM-based Cell Selection Criteria and Exact Prompt Template.

Criteria	Description
Cell Position	Specify the range or position of the cells you want to search. For example, you may want to search for explanations only in the cells of a specific column, row, or a particular section of the table.
Cell Content	Define the specific content or data type within the cells you want to search. For instance, you may want to search for explanations in cells containing numerical values, dates, specific keywords, or a combination of certain words.
Cell Formatting	Consider the formatting or styling applied to the cells. This could include searching for explanations in cells with bold or italic text, specific background colors, or cells that are merged or highlighted in a certain way.
Cell Context	Take into account the context surrounding the cells. You can search for explanations in cells that are adjacent to certain labels, headings, or identifiers, or within a specific context provided by other cells in the same row or column.
Cell Properties	Consider any specific properties associated with the cells. This might include searching for explanations in cells that have formulas, links, or other data validation rules applied to them.
Prompt	You will be given a parsed table **{Table}** in python dictionary format, extract the cells that need to be explained. The extraction rule should be based on the following criteria: **{Criteria}**. Only return the cells name in a python List[str].

C Metadata-based Augmentation

Metadata are defined as a form of formally represented background knowledge to understand the field semantics for correctly operating on table fields (or columns) and to further find common patterns in daily analysis (He et al., 2023). This analytical knowledge, particularly of field semantics, is able to increase the applicability across various tasks. In our table augmentation, we consider the following metadata:

(1) *Dimension / Measure*: This is one type of metadata used in Tableau (Hoelscher and Mortimer, 2018) and Excel (Ding et al., 2019) across diverse features. As the name suggests, the method involves categorizing each field in a table as either measure or dimension. The measure contains numerical data that can be subjected to calculations, such as the "Price" and "Discount". The dimension provides categorical information used for filtering, grouping, and labeling, such as the "Product Name" and "Category". Correctly classifying fields as either a measure or a dimension is crucial to determining feasible operations on the data and influences the accuracy and relevance of data analysis. (2) *Semantic Field Type*: Besides identifying whether a field is a measure or a dimension, semantic field type specifies the meaning and format of the data

within each field based on knowledge graphs. For example, the dimension field includes semantic field types such as "Consumer Product" and "Category", *etc*. Measure field includes semantic field types such as "Money" and "Ratio", etc. We follow the work (He et al., 2023) as a reference to this term. (3) *Table Size*: The size of a table is defined by its number of rows and columns. It provides essential context when determining the computational complexity of operations or understanding data density and granularity. (4) *Statistics Feature*: Statistics feature provides a quantitative representation of the tabular data. These features serve as numerical descriptors that summarize key aspects of the table datasets, aiding LLMs in understanding the overall characteristics and tendencies. Generally, statistics features include four categories (He et al., 2023): (a) Progression features (b) String features (c) Number range features (d) Distribution features, discussed in Section §4. We conducted empirical studies on common statistical features to identify the most appropriate combination for optimal utilization of TAP4LLM. (5) *Header Hierarchy*: Tables are often used to present data in a structured format, and headers play a crucial role in defining the meaning and context of the data in each column or row. The header hierarchy typically includes different levels of headers, each providing a level of organization and categorization for the data.

D Additional Experiment Settings

D.1 Downstream Tasks and Datasets

Table Reasoning Tasks. Each instance in table-based reasoning consists of a table T, a natural language question Q, and an answer A. Specifically, table T is defined as $T = \{v_{i,j} \mid i \leq \mathrm{R}_T, j \leq \mathrm{C}_T\}$, containing R_T rows and C_T columns. The content of the cell in the i-th row and j-th column is represented by vi, j. A question Q is a sequence of n tokens: $Q = \{q_1, q_2, q_3, \cdots, q_n\}$. In this paper, our primary focus is on two distinct table-based reasoning tasks, table-based fact verification (TFV) and table-based question answering (TQA). In TFV, the answer A is a boolean value in $\{0, 1\}$, indicating the veracity of the input statement (where 1 means the statement is entailed by the given table, and 0 means the statement is refuted by the given table). In TQA, the answer is a sequence of natural language tokens represented as $A = \{a_1, a_2, a_3, \cdots, a_n\}$ corresponding to the posed question. For our experiments, all tables first undergo table sampling and table augmentation by our proposed method and then are serialized into a sequence by table packing and serialization. Detailed implementation specifics are provided in Section §2.3.

In this paper, we mainly focus on tabular reasoning with two major tasks: TQA & TFV. We conduct experiments on five typical datasets and the distribution of the datasets can be found in Table 6. In addition, to extend our work to databases containing table structures, we also set up TAP4LLM on Spider (Yu et al., 2018) dataset. Specifically, we use: (1) **SQA** (Iyyer et al., 2017), which is constructed by decomposing a subset of a highly compositional dataset, WTQ (Pasupat and Liang, 2015). The dataset consists of 1,288 unique queries corresponding to 432 tables, with each table having 18.5 rows and 6.4 columns on average; (2) **HybridQA** (Chen et al., 2020b), which is designed as a large-scale multi-hop question-answering dataset over heterogeneous information of both structured tabular and unstructured textual forms. The dataset consists of 6,268 unique questions and each question is aligned with a Wikipedia table. Compared to the SQA dataset, HybridQA has shorter column numbers, which facilitates the understanding of the table's structure boundaries. (3) **ToTTo** (Parikh et al., 2020) is a high-quality English table-to-text dataset. It proposes a controlled generation task that involves synthesizing a one-sentence description given a Wikipedia table and a set of highlighted table cells. The dataset contains 8,026 samples, each comprising a Wikipedia table with highlighted cells. Each table contains 16 rows and 6 columns on average. (4) **FEVEROUS** (Aly et al., 2021) is a fact verification dataset over structured information. The dataset consists of 1,322 verified claims. Each claim is annotated with evidence in the form of sentences and cells from tables in Wikipedia. Each annotation also includes a label indicating whether the evidence supports, refutes, or does not provide enough information to make a decision. Each table contains 26.3 rows and 5.5 columns on average. (5) **TabFact** (Chen et al., 2020a) is another fact verification dataset where the tables are extracted from Wikipedia and the sentences are composed by crowd workers. Compared to the FEVEROUS dataset, TabFact encompasses a larger number of samples and each table has fewer rows, has 14 rows per table on average.

Metrics. For TQA and TFV tasks (SQA,

Table 6: The distribution of the used datasets.

Property	SQA	FEVEROUS	TabFact	HybridQA	ToTTo	Spider
Unique Query (Set Size)	1,228	1,322	9,228	6,268	8,026	10,181
Unique Table	432	942	1,342	4,364	5,934	500
SQL Query	-	-	-	-	-	5,693
Rows per tables (Median/Avg)	12 / 18.5	14 / 26.3	8 / 14.0	8 / 15.7	16 / 28.4	10 / 16.1
Columns per tables (Median/Avg)	4 / 6.4	4 / 5.5	4 / 5.5	4 / 4.3	6 / 8.8	4 / 4.5
Cells per tables (Median/Avg)	78 / 180.4	77 / 190.3	80 / 150.3	70 / 143.9	87 / 212.6	-
Domain	Wikipedia	Wikipedia	Wikipedia	Wikipedia	Wikipedia	-
Evaluation Metric	Exact Match	Exact Match	Exact Match	Exact Match	BLEU-4	Execution Accuracy

Table 7: Different kinds of table augmentation.

Knowledge Aspect	**Categories**	**Definition**
Dimension/Measure	Metadata-based	Distinguish each element in a table as either dimension field or measure field.
Semantic Field Type	Metadata-based	Classify the meaning and format of the data within each field based on knowledge graphs.
Table Size	Metadata-based	Basic information of a table including numbers of rows and columns.
Statistics Feature	Metadata-based	Statistics features such as change rate, numerical distribution, range of data.
Header Hierarchy	Metadata-based	The organization and structure of header elements within a table.
Docs References	Retrieval-based	External domain knowledge from reliable webpages (*e.g.*, wikipedia, Wolfram Alpha, *etc.*) which are similar to the given context.
Term Explanation	Retrieval-based	External domain knowledge such as term and metric definitions (formulas, relevant documents/sources, search results, etc.)
Self Prompting	Self-consistency-based	Leverage LLMs to generate some reasoning thoughts as supplementary for table augmentation (self-augmented prompting, chain-of-thoughts, *etc.*)

FEVEROUS, TabFact and HybridQA), we report the exact match accuracy of answer sets. For the data-to-text generation task (ToTTo), we report the BLEU-4 score.

D.2 Models

We evaluate the performance of the recent dominant LLM models, 1) Instruct-GPT-3.5 (Ouyang et al., 2022), using versions gpt-3.5-turbo, gpt-3.5-turbo-16k; 2) GPT-4, using the latest version of gpt-4 model; 3) Llama-2-70B (Touvron et al., 2023), using version 17; 4) Mixtral-8x7B (Jiang et al., 2024), using version 0.1.

Unless otherwise specified, we utilize **gpt-3.5-turbo** in all experiments. In the sampling methods, we use text-embedding-ada-002 (ope) for row and query embedding generation. The comparison experiments using other embeddings models, such as, text-search-ada-doc-001, bge-largen-en (Xiao et al., 2023), all-MinLM-L6-v2 (Reimers and Gurevych, 2019) can be found in Table 8. We set the temperature of all the models as 0, top p as 1.0, frequency penalty as 0, and presence penalty as 0.

The development of TAP4LLM begins with the foundation provided by LLMs. In designing our framework, we opt to use OpenAI models as our base model due to their excellent capabilities in language reasoning. However, the choice is not exclusive. Since TAP4LLM use natural language as an intermediary for interactive communication between the table and LLMs, it can also support other outstanding open-sourced models using natural language as input, such as Phoenix (Chen et al., 2023), ChatGLM (Zeng et al., 2022), Ziya (IDEA-CCNL, 2023), and Baichuan (Intelligence, 2023). This design provides versatility and flexibility in TAP4LLM implementation.

E Additional Experiments

E.1 Comparison Results of Embedding Type.

Based on the results from Table 8, we observe that: (1) *Superiority of "text-embedding-ada-002"*: "text-embedding-ada-002" consistently offers the best performance across the datasets. It suggests that for tasks similar to table reasoning, this embedding type might be the most suitable choice. (2) *Potential of "sentence-transformer"*: The "sentence-transformer" embedding type provides competitive results, especially in the ToTTo dataset. This suggests that it might be particularly suitable for certain tasks or datasets and is worth considering alongside "text-embedding-ada-002".

While "text-embedding-ada-001" and "bge-

Table 8: Comparative results of different embedding models on query-based sampling method without any augmentation method. We use all-MinLM-L6-v2 for the sentence-transformer. The highest performance of each dataset is bold.

Embedding Type	SQA	FEVEROUS	TabFact	HybridQA	ToTTo	Spider
text-embedding-ada-002	**28.32%**	**63.32%**	**59.80%**	**24.32%**	49.14%	**80.27%**
text-embedding-ada-001	27.12%	62.24%	57.32%	23.14%	48.21%	79.34%
bge-large-en (Xiao et al., 2023)	26.76%	62.87%	56.31%	22.65%	47.32%	78.25%
sentence-transformer (Reimers and Gurevych, 2019)	26.32%	63.31%	58.94%	23.78%	**50.12%**	80.05%

large-en" don't lead to the highest performance, they still provide competitive performance. This suggests that the choice of embedding can affect the overall performance, but the differences might not always be significant. The choice between these embeddings would likely depend on specific use cases, computational costs, and other practical considerations.

E.2 Comparison Results of Statistics Features

The accuracy of each dataset for four groups of statistics features reveals that the distribution features overall performed well in capturing the nuances and variations within specific tabular data entries. Based on this, we further propose a combination including the most practical features across these four categories and carry out an empirical study to examine its performance. Specifically, this combination contains variance, range, cardinality, major, and change rate. with each term's definition listed in Table 9. The experiment result, displayed in Table 10, demonstrates that our proposed combination surpasses the previous four feature sets across all six datasets.

Table 9: Detailed definition of statistics features.

Features	Definition
Progression Type:	
ChangeRate	Proportion of different adjacent values
PartialOrdered	Maximum proportion of increasing / decreasing adjacent values
OrderedConfidence	Indicator of sequentiality
String Features:	
AggrPercentFormatted	Proportion of cells having percent format
CommonPrefix	Proportion of most common prefix digit
CommonSuffix	Proportion of most common suffix digit
Number Range Features:	
Aggr01Ranged	Proportion of values ranged in 0-1
Aggr0100Ranged	Proportion of values ranged in 0-100
AggrIntegers	Proportion of integer values
AggrNegative	Proportion of negative values
Distribution features:	
Variance	Standard deviation of a given series of data
Range	Values range
Cardinality	Proportion of distinct values
Spread	Cardinality divided by range
Major	Proportion of the most frequent value
Benford	Distance of the first digit distribution to real-life average
Skewness	Skewness of numeric values
Kurtosis	Kurtosis of numeric values
Gini	Gini coefficient of numeric values

Table 10: Comparative results of various types of statistical features. The experiment setting is the same as Section 2. The highest performance of each dataset is bold.

Statistics Features Type	SQA	FEVEROUS	TabFact	HybridQA	ToTTo	Spider
Progression features	29.20%	64.26%	60.45%	25.11%	49.53%	77.47%
String features	28.56%	63.13%	61.38%	24.83%	48.29%	73.56%
Number range features	29.13%	62.18%	59.03%	24.53%	49.68%	76.32%
Distribution features	30.28%	66.34%	62.18%	24.76%	49.34%	79.14%
Statistics features	**31.22%**	**66.51%**	**62.33%**	**26.13%**	**50.57%**	**80.94%**

E.3 TAP4LLM in Open-source model

Beyond conducting experiments on GPT models. we also evaluate the effectiveness of TAP4LLM on two recent LLMs: Llama-2-70B and Mixtral-8x7B. According to Table 11, we first evaluated direct inference on open-source models and then apply TAP4LLM to each model. The result demonstrates that TAP4LLM increases models' performance on all five datasets.

In addition, TAP4LLM is currently evaluated under the setting of in-context learning. However, through parameter-efficient fine-tuning (Hu et al., 2021) and recently advanced prompt compression techniques (Jiang et al., 2023a; Zou et al., 2024), we can directly apply TAP4LLM on more challenging tabular reasoning tasks requiring training procedures. We will leave this as our future work.

Table 11: Comparison of TAP4LLM and baseline on Open-source models. We refer to "Baseline" as directly inferring each task using the model. For TAP4LLM, we apply semantic sampling for table sampling module and Statistics Feature/D/M+SF/self-prompting for table augmentation module.

Model Name	Methods	SQA	FEVEROUS	TabFact	HybridQA	ToTTo
Llama-2-70B	Baseline	19.02%	65.33%	63.45%	17.21%	21.08%
	TAP4LLM	22.14%	69.20%	66.32%	23.15%	30.00%
Mixtral-8x7B	Baseline	21.25%	61.32%	57.21%	21.01%	34.25%
	TAP4LLM	24.18%	63.29%	58.80%	25.44%	37.79%

E.4 TAP4LLM in Database Application

Dataset We test TAP4LLM effectiveness on Spider (Yu et al., 2018). Spider is a cross-domain Text-to-SQL dataset as shown in Table 6. Each instance contains a natural language question, a specific database containing tabular information, and one corresponding SQL query.

Metric We evaluate TAP4LLM on the development split *Spider-dev* which contains 1034 instances over 200 databases. We use the Execution Accuracy, followed by the original paper (Yu et al., 2018), to compare the execution output of the predicted SQL query with the golden SQL query.

Experiment As shown in Table 12 and Table 13, the experiment result demonstrates that LLMs

achieve an overall higher model performance through TAP4LLM. Specifically, the execution accuracy reaches the highest through semantic-based sampling and D/M + SF augmentation.

Table 12: Comparative results of the table sampling methods on Spider.

Sampling Type	Table Sampling Methods	Execution Accuracy
Rule-based Sampling	Random Sampling	74.58%
	Evenly Sampling	72.03%
	Content Snapshot (Yin et al., 2020)	78.93%
Embedding-based Sampling	Centroid-based Sampling	77.43%
	Semantic-based Sampling	80.27%
	w/ Column Grounding	**81.03**%
	Hybrid Sampling	78.94%
LLM-based Sampling	LLM-Decomposer (Ye et al., 2023b)	78.34%
-	No sampling (GPT-3.5)	72.15%
	No sampling (GPT-3.5, truncated)	68.47%

Table 13: Comparative results of table augmentation methods on Spider. We use semantic-based sampling method without augmentation as the default method for table augmentation.

Augmentation Aspect	Execution Accuracy
Baseline	80.27%
D/M + SF	**82.45**%
Statistic Feature	80.94%
Term Explaination (LLM-based)	80.48%
Term Explaination (Heuristics-based)	80.33%

F Implementation Details

F.1 Motivation of our Framework

Table Sampling: One primary challenges for tabular reasoning is that the full content of a table could be very long and noisy to be include in the prompt. Most LLMs have a limited input context window size (*e.g.*, 4k tokens) in which an overlong table cannot fit it. For long tables that satisfy the length constraint, it can still lead to unnecessary computations (of LLMs on long prompt) and quality regressions (generation interfered by noisy input) when placing irrelevant table content (*w.r.t.* the task or query) in the prompt.

Table Augmentation: Another challenge is what additional/external knowledge could help LLMs better understand a table? The raw content of a table may contain ambiguous information (*e.g.*, abbreviations, domain-specific terms, column type, *etc*) that requires further interpretation and clarification. We are motivated to propose table augmentation for 1) *enhanced contextual understanding*: by supplementing tables with metadata and attributes, we can achieve a more profound grasp of the table's intrinsic structure and semantics and further enrich the tabular data; 2) *bridging external knowledge gasps*: tables alone might not encompass all the required information to provide comprehensive answers to certain queries. By retrieving external knowledge from reliable sources, *e.g.*, Wikipedia, we can aid the language models in understanding the broader context of the query, leading to more informed and nuanced responses.

Table Packing: The desire to maintain efficient reasoning without changing the LLMs architecture motivates us to consider how to encode the table into a prompt? While sampling and grounding compress the table content, augmentation expands the prompt by adding more information. With a given token budget, one needs to find the balance to allocate available tokens between table content and augmented knowledge.

F.2 Table Syncing

To achieve the interactive table reasoning, TAP4LLM proposes the "table sync" to ensure that applications, such as Excel Copilot, maintain their table data in synchronization with the table manager. The table manager acts as a go-between, managing the data that is either stored locally in a cache or accessed through a database connection. Specifically, when changes are made to the data within the application, those changes must be reflected in the table manager for any operation performance, such as sampling, augmentation, and packing. Conversely, if changes are made within the table manager, the changed data should be updated in the application as well.

This syncing process is essential for maintaining data integrity and ensuring that all components of the system are kept up-to-date. This is especially beneficial when the data is being used to generate prompts for a large language model, as it allows for accurate data processing, querying, and analysis. By having the most current and relevant information, the model can provide accurate and reliable responses.

F.3 Table Cleansing

Table cleansing is an independent step in tabular data prepossessing, especially when dealing with hierarchical tables. In the context of fine-grained in-context learning, where pre-trained generated model has to discern and process intricate patterns and relationships within datasets. The importance of clean and standardized tables cannot be overstated for two reasons: (1) Dirty or unorganized

tabular data can mislead the models and impair the model's performance; (2) Cleansed tables ensure uniformity, making them easier to compare, merge, or use in subsequent operations. For example, imagine a financial analyst case aiming to forecast a company's stock price based on historical data. The corresponding table contains daily stock prices, trading volumes, and various financial indicators. If there are any missing certain values for certain days, or duplicate entries due to system glitches. Such inconsistencies may dramatically affect the forecasting performance. For instance, it might suggest a non-trading day or a sudden drop in stock price. Specially, the formal definition of table cleansing is: Given a table T consisting of rows R_T and columns C_T, table cleansing transforms T into T' such that: (a) *Cell and column name completeness*: For every cell $c_{i,j}$ in T where $i \in R_T$ and $j \in C_T$, if $c_{i,j}$ has a missing or null value, it is filled using contextual information (*i.e.*, use the corresponding entire column C_j of cell $c_{i,j}$ as the context). We utilize a separate "CallLLM" system $g(\cdot)$ to call a pre-trained language model to synthesize the missing value. The processing can be formulated as $c_{i,j} = g(C_j)$. This ensures that gaps in the data don't lead to misleading interpretations or missed patterns. (b) *Duplicate data points removal*: For every pair of rows r_m, r_n and pair of columns c_p, c_q in T, if $r_m = r_n$ or $c_p = c_q$ respectively, one from the pair is removed to eliminate duplication. (c) *Format consistency*: For every cell $c_{i,j}$ in T, the value conforms to a specific format, unit, or pattern.