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

Yuan Sui*
National University of Singapore
yuan.sui@u.nus.edu

Jiaru Zou*
University of Illinois Urbana-Champaign
jiaruz2@illinois.edu

Mengyu Zhou†
Microsoft
mezho@microsoft.com

Xinyi He*
Xi'an Jiaotong University
hxyhxy@stu.xjtu.edu.cn

Lun Du
Microsoft
lun.du@microsoft.com

Shi Han, Dongmei Zhang
Microsoft
{shihan,dongmeiz}@microsoft.com

ABSTRACT

Table reasoning has shown remarkable progress in a wide range of table-based tasks. These challenging tasks require reasoning over both free-form natural language (NL) questions and semi-structured tabular data. However, previous table reasoning solutions suffer from significant performance degradation on "huge" tables. In addition, most existing methods struggle to reason over complex questions since they lack essential information or they are scattered in different places. To alleviate these challenges, we exploit a table provider, namely TAP4LLM, on versatile sampling, augmentation, and packing methods to achieve effective semi-structured data reasoning using large language models (LLMs), which 1) decompose raw tables into sub-tables with specific rows/columns based on the rules or semantic similarity; 2) augment table information by extracting semantic and statistical metadata from raw tables while retrieving relevant knowledge from trustworthy knowledge sources (e.g., Wolfram Alpha, Wikipedia); 3) pack sampled tables with augmented knowledge into sequence prompts for LLMs reasoning while balancing the token allocation trade-off. We show that TAP4LLM allows for different components as plug-ins, enhancing LLMs' understanding of structured data in diverse tabular tasks.

PVLDB Reference Format:
Yuan Sui, Jiaru Zou, Mengyu Zhou, Xinyi He, Lun Du, and Shi Han, Dongmei Zhang. TAP4LLM: Table Provider on Sampling, Augmenting, and Packing Semi-structured Data for Large Language Model Reasoning. PVLDB, 14(1): XXX-XXX, 2024.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://anonymous.4open.science/r/TableProvider-4CC3.

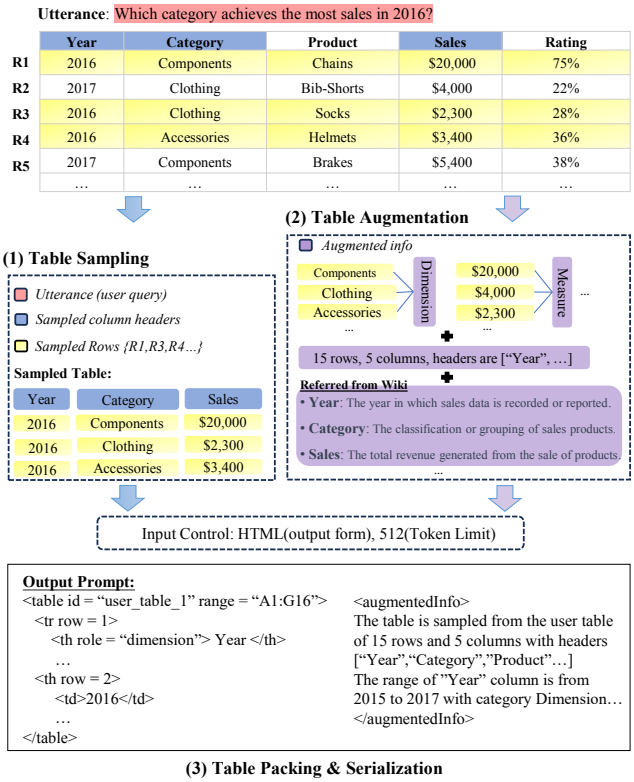

Figure 1: Demonstration of the Three TAP4LLM Modules. (1) Table sampling: sample most relevant content. (2) Table augmentation: retrieve and add extra information. (3) Table packing: serialize the sampled table and augmented information into a string while controlling the token allocation.

1 INTRODUCTION

The extensive and complex characteristics of data are commonly represented in the format of structured data. It makes the data more manageable and facilitates data analysis and processing by machines. **Table** is one of those fundamental and widely used semi-structured data types in relational databases, spreadsheet applications, and programming languages that handle data for various domains, including financial analysis [38, 69], risk management [4],

*Yuan, Jiaru and Xinyi made their contributions during their internship at Microsoft.
†Corresponding author.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

healthcare analytics [57], *etc*. Reasoning over tabular data is a fundamental aspect of natural language understanding (NLU) and information retrieval (IR), and has several downstream tasks, such as Table-based Question Answering (TQA) [9, 11, 29, 64], Table-based Fact Verification (TFV) [8, 17, 63], Table-to-Text [60], Text-to-SQL [66], Column Type & Relation Classification [12, 27],*etc*.

In the field of natural and programming language understanding and generation, LLMs (large language models) have demonstrated impressive abilities as few-shot reasoners with prompt engineering and in-context learning [33, 34]. Their capacity to generate human-like text and human-level code [11, 14, 64] has opened up new possibilities for reasoning over tabular data [7, 15]. However, it is non-trivial to directly utilize vanilla LLMs such as GPT [40] to consume tables as their prompt inputs.

First, which part of a table should be kept in the prompt? The full content of a table could be very long and noisy to be included in the prompt. Most LLMs have a limited input context window size (*e.g.*, 4k tokens) in which an overlong table cannot fit. For long tables that satisfy the length constraint, it can still lead to unnecessary computations (of LLM 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. To address the challenge, some table sampling methods were proposed in ad-hoc ways. For example, truncating the input tables to contain only the first 22 rows and 8 columns [7], or filtering relevant rows based on n-gram overlap between them and the utterance [65]. To answer the question of which part to keep, we need a more systematic study of different grounding and sampling algorithms.

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 on the raw tables may lead to misinterpretation and bias in the LLMs' outputs. To address this, some augmentation techniques were proposed to incorporate structured knowledge [54, 63], common sense knowledge [5, 18, 44, 52], and analytical knowledge [19, 30] into pre-training and inference processes. For example, [30] transforms existing tabular data to create diverse natural language inference instances for better zero-shot performance. [19] infers implicit metadata behind raw table content through field distribution and knowledge graph information. However, the techniques were proposed independently and there lack of comprehensive study that compares them and tries to combine them to provide useful and diverse knowledge and thoughts for LLMs.

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 (as discussed in our previous study [54]), but also determines the string length of the serialized table and the augmented information. For example, as discussed in [54], table formats such as HTML [3] 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.

In this paper, we propose **TAP4LLM** (table provider for large language models) as a versatile pre-processing toolbox to generate table prompts in LLM reasoning. TAP4LLM addresses the above challenges with three corresponding modules (i) Table Sampling: Selecting a sub-table T' from the raw table T based on the rules or semantics of the query or utterance Q; (ii) Table Augmentation: Enhancing T' by integrating relevant external knowledge, metadata, and attributes based on the raw Table T; and (iii) Table Packing: Packing the sampled table T' with the augmented knowledge into a sequence with a specified serialization format for LLMs while balancing the token allocation trade-off. In each module, we collect and design many different methods for usage in various scenarios (*e.g.*, speed over accuracy). Across six distinct datasets, our findings demonstrate that TAP4LLM significantly improves accuracy by achieving an average enhancement of 6.02% through the table sampling module, 3.29% through the table augmentation module, and 1.38% through the table packing module. Collectively, TAP4LLM elevates accuracy by an average of 7.93% when compared to the direct input of raw tables into LLMs.. Our exploration has led us to conclude that TAP4LLM enhances the interaction between LLMs and tabular data by employing effective pre-processing.

In addition, through multiple empirical studies, we develop a comprehensive guideline for the effective utilization of TAP4LLM. Our investigation reveals that the optimal performance of sampling is achieved through the query-based method. We then conducted evaluations on augmentation methods and proposed the most suitable combination of sampling and augmentation for each dataset. Ablation studies are also conducted on token allocation to identify trade-offs between model performance and resource usage.

In summary, our main contributions are:

- We proposed a unified pre-processor TAP4LLM to improve the effectiveness of LLMs in tabular reasoning tasks. Our framework includes three modules: table sampling, table augmentation, and table packing.
- We conducted a comprehensive survey to evaluate the effectiveness of TAP4LLM. Through various sampling, augmentation, and packing methods, TAP4LLM achieves an average of improved performance by 7.93%. We also assessed the effectiveness of different configurations inside TAP4LLM, including embedding types and statistical features.
- We formulated a comprehensive usage guideline for our framework TAP4LLM. First, we identify the optimal combination of methods within TAP4LLM for different usage scenarios. Second, we provide a trade-off map between performance metrics and token allocation.

2 PRELIMINARIES

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

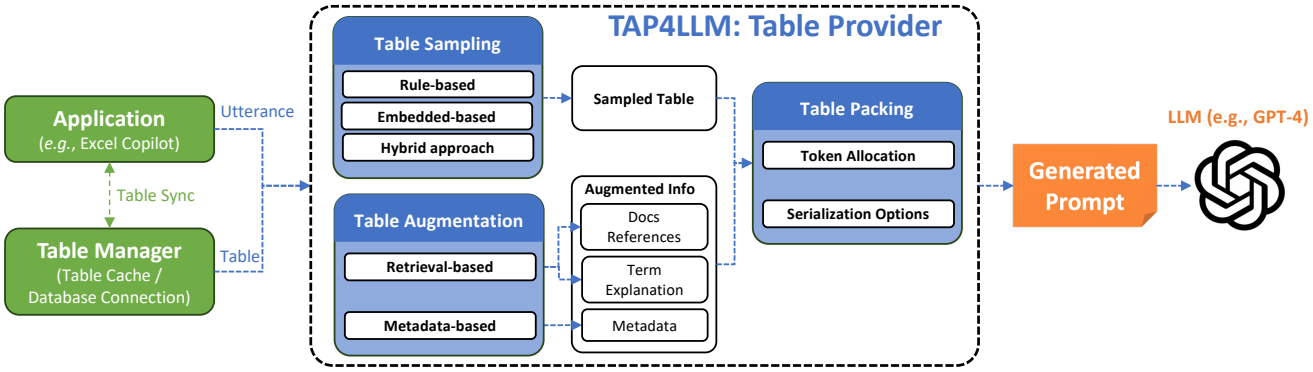

Figure 2: TAP4LLM Framework for Tabular Data. Noted that the "table sync" refers to the application (such as Excel Copilot) keeps its table data in sync with the table manager. The table manager acts as an intermediary, managing the data that is either stored locally in a cache or accessed through a database connection. This syncing process is crucial for "interactive table reasoning" and for maintaining data integrity. The implication of this syncing process is further discussed in §5.

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. Note that various serialization methods have already been explored in our previous work [54]. Detailed implementation specifics are provided in Section §3.3.

Challenges with Long Sequences in LLMs. Most LLMs struggle with long sequences due to the quadratic complexity of self-attention mechanisms [55, 58]. It's important to note that the maximum sequence length of commonly used LLMs, such as gpt-3.5-turbo-16k [2], is limited to 16k tokens. This limit becomes especially problematic when dealing with structured data, which typically consists of various components and causes significant memory and computational challenges. Solutions like Tapex [39] and Tapas [20] utilize a simple truncation approach based on the table's sequence length. However, these methods exhibit the shortcomings of excluding important data and disrupting the whole table structure. In our work, we introduce specific constraints for the LLMs' call requests. For example, to avoid structural disruptions from truncation, we use a *"token allocation"* strategy to estimate the token capacity and initiate the sampling functions (explained in Section §3.1) when the table's token count surpasses a certain threshold.

Neuro-Symbolic Functions & Metadata Analysis. Neuro-symbolic method is a specialized approach in Deep Learning (DL) that combines neural and symbolic methodologies [21]. In this work, we use Metadata Analysis [19], a neuro-symbolic method based on a transformer architecture [58], to generate metadata essential for table analysis. Such metadata serves as an explicit representation of formally organized background knowledge. An example of this is the binary metadata classification for column types, known as dimension and measure[1]. Furthermore, various LLM-centric toolsets have been developed to facilitate: (i) integration of LLMs with diverse data sources, and (ii) dynamic interactions of LLMs with their surroundings. We select some of symbolic functions from the toolsets as our agents for table augmentation, including Langchain [35], Semantic Kernel [42], *etc.*

3 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**: Explicitly incorporate relevant external knowledge, metadata, and attributes about the original table T.
- **Table Packing**: Control the token allocation for table sampling and table augmentation; Convert the structured table into a sequence (table serialization).

3.1 Table Sampling

One of the primary challenges for reasoning over tabular data with LLMs is *(1) token efficiency*. Due to the inherent structure of table(s), even a modest-sized table can quickly consume a significant portion of the available token budget of the LLMs. While study[54] shows that representing tables in formats like *HTML* [3] or *XML* can improve the performance of GPT models, it also leads to increased token usage. Furthermore, *(2) noisy information* becomes an issue in large tables, especially those with over 30 rows, and becomes

[1]A measure is a table field consisting of numerical data that you can calculate, such as 'Price' and 'Discount'. Conversely, a dimension is a field that offers categorical information used for filtering, grouping, and labeling, such as the 'Product Name' and 'Category'. It's important to correctly classify fields as measures or dimensions because they dictate the operations you can perform on the data, affecting the accuracy and relevance of the analysis.

worse when the row count exceeds 60 [11, 64]. Such noise can obscure the primary intent of a query, leading to potentially skewed or irrelevant model outputs. Even with advancements like GPT-4[2], which can handle up to 32k tokens, the challenges caused by noisy information remain [14]. In this context, although more and more LLMs have been developed to handle longer sequences, the complexity of reasoning over tabular data highlights the need for sampling techniques. These approaches focus on the most relevant part of a table, providing a practical solution to bridge the gap between the nature of tables and the context boundaries of LLMs.

In table sampling, a subset of top-ranked rows and columns is selected to form the sub-table. Specifically, given an original table T with a distinct set of rows R_T, columns C_T, and a query q, the goal of table sampling is to produce a sub-table $T' = T_{r,c}$, where $r \in \mathcal{P}(R_T)$, $c \in \mathcal{P}(C_T)$. Here $\mathcal{P}(X)$ denotes the power set of X, representing all possible subsets of X. The table sampling formulation can be represented as follows:

$$T' = T_{r,c} = select(T, rank(f(T, q))) \tag{1}$$

The $f(T, q)$ function represents each sampling method. For example, the query-based sampling (discussed in detail below) calculates the similarity score as f between the query q and each row/column from T. The $rank()$ function sorts the rows and columns of T based on sampling methods f and outputs a ranked list. The $select()$ function then chooses the top-k rows and top-l columns from the ranked list to form the sub-table $T_{r,c}$. Note that, in the context of column-based sampling, the term "grounding" is frequently used. Grounding becomes essential when dealing with ambiguous queries. For instance, the term "date" could refer to calendar data, a romantic outing, or the fruit. If a table includes columns like "release date", "date of birth", or "date ate", grounding helps determine which column(s) the query refers to (See the improvement of using column grounding in Table 4). Specifically, we propose two distinct variants for table sampling as follows:

3.1.1 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. We consider three common rule-based sampling methods as follows:

(1) *Random Sampling*: Random sampling refers to selecting rows from a table, with each having an equal probability of being selected. This ensures that the sampled subset is diverse and unbiased, as it reflects the characteristics of the table without any distortion. Even though random sampling may not always capture the full diversity or variations in the entire table, it remains a practical choice for table sampling implementation, as directly supported by various statistical tools and software.

(2) *Evenly Sampling*: To avoid overwhelming excessive data while ensuring a balanced and comprehensive subset, we utilize evenly sampling as one of the table sampling variants. It allows for systematic sampling of rows or columns from the table in a specific sequence. Specifically, evenly sampling starts with the topmost field of the table, T, and alternates between the beginning (r_1) and the end of the table (r_n, where the table has n rows) to create a subset $r \in \mathcal{P}(R_T)$. This alternating pattern continues from the outermost rows towards the middle until the aggregated number of tokens from the selected rows reaches a pre-defined token threshold.

(3) *Content Snapshot & Synthetically Sampling*: Table contents provide more details about the semantics than column headers themselves. When taking the utterance into consideration, the most relevant cell values to answer the utterance might come from multiple rows. we follow this work [65] to create the method of content snapshots for table K rows based on their relevance to the utterance along with a synthetic strategy. Specifically, for $K > 1$, top-K rows are selected based on the highest n-gram overlap ratio with the utterance. 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. For instance, consider the utterance "How many more participants were there in 2008 than in the London Olympics?", and an associating table with column names "Year", "Host City", and "Number of Participants". The most relevant cells to the utterance, "2008 (from Year)" and "London (from Host City)", come from different rows. Through the synthetic strategy, these insightful contents can be combined into one single synthetic row and encoded as a subset of table content most relevant to the utterance.

3.1.2 Embedding-based Sampling. Embedding-based sampling is an advanced approach to table sampling based on high-dimensional vector embeddings. 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. By mapping each row or column to vectors, this method harnesses the power of spatial relationships within the embedding space to guide sampling decisions. Specifically, we propose two embedding-based sampling methods as follows:

(1) *Query-based Sampling*: Query-based Sampling is a tailored approach emphasizing the semantics relevance of table cells to the utterance. The process is exactly illustrated in Eq. 1. Note that the default query-based sampling is the row-based method. We also study the significance of column grounding in table sampling by testing query-based sampling in conjunction with column grounding, as shown in Table 3.1. By incorporating column grounding, the sampling process revolves around both rows and columns, offering a dynamic table subset where both rows and columns are susceptible to change based on the user query.

(2) *Clustering-based Sampling*: In Clustering-based Sampling, the core principle is to group similar rows or columns based on their embedding representation. 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. The goal of clustering-based sampling is to extract a sub-table $T' \subseteq T$ and ensure the preservation of data diversity. We use *K-Means* [41] as our clustering-method backbone. Specifically, for each $r \in R_T$ or $c \in C_T$, compute its embedding as $e = E(r)$ or $e = E(c)$. Then, utilize the K-Means algorithm, and partition the set of embeddings into n clusters: $\{C_1, C_2, \ldots, C_n\}$. For each cluster, C_i, select the top-K rows or columns based on a specified criterion (*e.g.*, closeness to centroid). At last, aggregate the chosen rows or columns from all clusters to form T'.

[2]gpt-4-32k: maintains the same capabilities as standard GPT-4 mode but has a 4x context length, equating to 32,768 tokens. More details can be found at https://platform.openai.com/docs/models/gpt-4.

Table 1: 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.*)

3.2 Table Augmentation

Table augmentation has emerged as a crucial approach in navigating the challenges associated with the intricate and varied nature of tabular data. Two primary benefits arise from this approach. **Firstly**, *Enhanced Contextual Understanding*: Raw tables often come without ample context or explicit metadata, such as distribution or dimension/measure categorizations for columns. 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. **Secondly**, *Bridging External Knowledge Gaps*: 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. (See the various knowledge aspects used in TAP4LLM from Table 1.)

3.2.1 Metadata-based Augmentation. Metadata is defined as a form of background knowledge to understand the field semantics for correctly operating on table fields (or columns) and to further find common patterns in daily analysis [19]. This analytical knowledge, particularly of field semantics, is able to increase the applicability across various tasks. Specifically, we consider the following metadata-based augmentation types:

(1) *Dimension / Measure*: This is one type of metadata used in Tableau [22] and Excel [13] 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 [19] 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 [19]: (a) Progression features (b) String features (c) Number range features (d) Distribution features, discussed in Section §6. 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.

3.2.2 Retrieval-based Augmentation. Large Language Models have occasionally been observed to generate hallucinated or factually inaccurate text [70, 72]. To mitigate these issues, several works have proposed to strengthen LLMs with information retrieval systems (retrieval-augmented LLMs) [31, 43, 53], which enables LLMs to retrieve relevant contents 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 [43]. 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 [31, 43, 53] suggest, 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: **(i) document references**: to provide supplemental relevant web pages as the

references for the given table; **(ii) term explanation**: to explain strange/ambiguous term in the given table. We utilize technologies like vector databases [59] and LangChain [35] to facilitate the retrieval of pertinent information from Wikipedia[3].

Table 2: 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].

(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 [35] 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.

(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: *i) 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 2. *ii) Heuristics-based Cell Selection*: Inspired by the methodology presented in [20], 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. The comparative experiment results of these two variants can be found in Table 6.

3.2.3 Self-consistency-based Augmentation. We follow [54] to provide the self-consistency-based augmentation approach. First, we append the instruction "Identify critical values and ranges of the last table related to the statement" to the initial prompt, and then forward this prompt to the LLM. The output generated from this instruction is then incorporated back into the prompt. Following this, we reintroduce the enriched prompt, now containing both the initial query and the newly generated insights, to the LLM along with the task-specific instructions for further processing. This iterative approach, namely *self Prompting*, allows for a deeper contextual understanding and analysis by LLMs. Formally, self-prompting is a simple idea that utilizes prompting twice to leverage the capabilities of LLMs in understanding structured data. In essence, through this self-consistency-based augmentation, we seek to harness and amplify the latent structured data reasoning prowess of LLMs.

3.3 Table Packing

Table Packing takes the pre-processed table T as the input and returns table serialization and packing function to convert a table into a sequence. One of the significant limitations of current LLMs, besides, hallucination and instability, is their severe input limits. Most LLMs can only handle $4k$~$16k$ tokens, as seen in models like GPT-3.5 [6]. Some selected LLMs like GPT-4 [46] can process up to $32k$ tokens input. However, this limitation becomes apparent when processing large tables, which often exceeds these token limitations by a substantial margin. Recently, Anthropic's Claude 2[4], a newer generation of LLMs, boasts significantly higher token limitations of up to 100,000 tokens. The advancement allows up to input the entire table to the LLM all at once. However, the performance of these models scales inversely with prompt size and happens across most LLMs architectures. The desire to maintain efficient reasoning without changing the LLMs architecture motivates us to propose the *token allocations* module. The packing component supports token-limit allocation for table sampling and augmentation. We

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

[4]https://www.anthropic.com/index/claude-2

Table 3: 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	Wikipedia
Evaluation Metric	Exact Match	Exact Match	Exact Match	Exact Match	BLEU-4	Exact Match

conduct an empirical study to determine the proper proportion of the sub-table length and augmentation information length, as shown in Figure 3. The packing process is controlled by a user-defined parameter token limit, which determines the maximum truncate token length.

Moreover, our empirical study [54] emphasizes a noteworthy observation regarding using markup languages like *HTML* or *XML* leads to much better generation quality by LLMs over TQA and TFV. In this pattern, TAP4LLM support multiple serialization functions, *e.g.*, HTML, XML, JSON, CSV, NL+Sep (one of the typical options, *e.g.*, using '|' as cell/column separator) and Markdown,*etc.*

4 EXPERIMENTS

4.1 Experiment Settings

4.1.1 Downstream Tasks and Datasets. 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 3. In addition, to extend our work to databases containing table structures, we also set up TAP4LLM on Spider [66] dataset. Specifically, we use: (1) **SQA**, which is constructed by decomposing a subset of a highly compositional dataset, WTQ [48]. 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**, 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** 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** 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** 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. (6) **Spider** is a semantic parsing text-to-SQL dataset across multiple domains. The dataset consists of 10,181 questions and 5,693 unique complex SQL queries on 200 databases with multiple tables.

4.1.2 Models. In this study, we evaluate the performance of the recent dominant LLM models, 1) Instruct-GPT-3.5 [47], using versions gpt-3.5-turbo, gpt-3.5-turbo-16k; 2) GPT-4, using the latest version of gpt-4 model. Unless otherwise specified, we utilize **gpt-3.5-turbo** in all experiments. In the sampling methods, we use text-embedding-ada-002 [1] for row and query embedding generation. The comparison experiments using other embeddings models, such as, text-search-ada-doc-001, bge-largen-en [62], all-MinLM-L6-v2 [51] can be found in Table 7. 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 LLaMa [56], Phoenix [10], ChatGLM [67], Ziya [26], and Baichuan [28]. This design provides versatility and flexibility in TAP4LLM implementation.

4.2 Comparison Results of Table Sampling

According to Table 4, we conduct the comparative experiments on multiple table sampling methods and make several observations as follows:

(1) *Effectiveness of query-based and clustering-based Sampling*: The query-based sampling method, combined with column grounding, achieves the best performance over all datasets. It indicates that sampling specific, relevant parts of the table that are relevant to the query can enhance LLMs' capability to produce more accurate results. Additionally, the use of both row-based and column-based sampling can provide a fine-grained subset of the table that covers the most essential information for the reasoning process. Similarly, the clustering-based sampling method demonstrates satisfactory performance. This method uses the patterns and similarities within the data point in each table to create the clusters that guide the sampling process. It doesn't consider the relevance between the table

Table 4: 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.

Table Samping Methods	SQA	FEVEROUS	TabFact	HybridQA	ToTTo	Spider
Clustering-based Sampling	28.10%	63.50%	55.40%	24.03%	48.30%	77.43%
Query-based Sampling	28.32%	63.32%	59.80%	24.32%	49.14%	80.27%
w/ Column Grounding	**29.12%**	**64.74%**	**60.23%**	**25.14%**	**53.42%**	**81.03%**
Random Sampling	27.30%	60.30%	55.17%	23.60%	40.12%	74.58%
Evenly Sampling	26.72%	61.87%	54.63%	5.32%	29.41%	72.03%
Content Snapshot	28.24%	63.10%	56.92%	23.40%	47.51%	78.93%
No sampling (GPT-3.5)	27.60%	60.12%	56.20%	14.10%	47.42%	72.15%
No sampling (GPT-3.5, truncated)	23.54%	43.54%	52.12%	23.12%	30.42%	68.47%

Table 5: Comparative results of the table augmentation methods. We use query-based sampling method (no augmentation) as the baseline. The term "Delta" refers to the performance gap between each method and the baseline. The top-3 performance gap on each dataset are highlighted in green, with the best performance being underlined. Noted that since only the Totto dataset contains hierarchical headers, we only provide the "header hierarchy" method on this dataset.

Table Augmentation Methods	SQA		FEVEROUS		TabFact		HybridQA		ToTTo		Spider	
	Acc	Delta	Acc	Delta	Acc	Delta	Acc	Delta	BLEU-4	Delta	Acc	Delta
baseline	28.32%	0.00%	63.32%	0.00%	59.80%	0.00%	24.32%	0.00%	49.14%	0.00%	80.27%	0.00%
Dimension/Measure+ Semantic Field Type	30.12%	1.80%	65.72%	2.40%	**62.67%**	2.87%	26.12%	1.80%	51.25%	2.11%	**82.45%**	2.18%
Table Size	28.85%	0.53%	63.40%	0.08%	60.30%	0.50%	24.94%	0.62%	49.03%	-0.11%	79.86%	-0.41%
Statistics Feature	31.22%	2.90%	**66.51%**	3.19%	62.33%	2.53%	26.13%	1.81%	50.57%	1.43%	80.94%	0.67%
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%	77.65%	-2.62%
Term Explanations	31.59%	3.27%	64.12%	0.80%	62.32%	2.52%	26.24%	1.92%	**53.21%**	4.07%	80.48%	0.21%
Self Prompting	30.45%	2.13%	65.24%	1.92%	62.32%	2.52%	**26.64%**	2.32%	52.36%	3.22%	-	-

and the query. Instead, it mainly focuses on the inner information of the table across different clusters.

(2) *Potential of Content Snapshot*: "Content Snapshot [65]" shows result that is relatively close to "Clustering-based Sampling" and "Query-based Sampling". It indicates that capturing a snapshot of essential information from the table can provide a useful cross-section for the LLMs to learn from. Even though it might not always capture the most relevant or critical information that is needed for table reasoning, it demonstrates its potential as a viable sampling technique. Furthermore, compared to the query-based or clustering-based sampling, this method doesn't require time-consuming embedding generation. Instead, it calculates the n-gram overlap with the utterance and the process is linear with the number of rows.

(3) *Table Sampling vs. Direct Encoding (Truncation)*: Directly leveraging the GPT-3.5-turbo model with 32k tokens limitation shows worse performance compared to using table sampling methods. Even though it can cover more information on the table, however, the results demonstrate that encoding much of the table information might introduce noise and disrupt the table reasoning process. Furthermore, using a 4K token-sized context window model with truncation results in even poorer performance compared to the model with a 32k token limitation. It indicates that truncating the table could lead to the loss of vital context or information, potentially restricting the model's capability to fully understand the table and generate comprehensive answers.

4.3 Comparison Results of Table Augmentation.

For the comparative experiments of table augmentation methods, we use the query-based sampling method as the baseline method and report the performance gap between each augmentation method and the baseline. According to Table 5, several insights can be found as follows:

(1) *Effectiveness of table augmentation:* From the experimental results, we found that table augmentation methods further improve LLM's reasoning ability after sampling. For example, "Dimension/Measure + Semantic Field Type" achieves higher accuracy across all six datasets (most significant increase on TabFact +2.87%). "Statistics Feature" performs effectively on datasets that include a higher proportion of statistical cell contents (FEVEROUS +3.19%). "Docs References" and "Term Explanations" add meaningful context and semantic understanding to the model's processing of tables, with (SQA +5.13%, ToTTo +4.07%). Also, "self-prompting" helps the model generate better queries or responses by encouraging it

Table 6: Comparative results of cell selection functions. We use query-based sampling as the baseline.

Methods	SQA		FEVEROUS		TabFact		HybridQA		ToTTo		Spider	
	Acc	Delta	Acc	Delta	Acc	Delta	Acc	Delta	BLEU-4	Delta	Acc	Delta
LLM-based	31.59%	**3.27%**	64.12%	**0.80%**	62.32%	**2.52%**	26.24%	**1.92%**	53.21%	**4.07%**	80.48%	**0.21%**
Heuristics-based	29.59%	1.27%	63.72%	0.40%	61.58%	1.78%	25.24%	0.92%	51.21%	2.07%	80.33%	0.06%

to use its own outputs as a guide for further response generation (ToTTo +3.22%). We also found some methods might not improve the model's performance by adding extra information. According to the result, the model performance with "Table Size" is minimal, suggesting little help in adding substantial semantic information. In addition, the result of "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) *Comparison of different cell selection methods:* For the "Term Explanations" method, we compare two distinct cell selection functions as shown in Table 6. 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).

(3) *Combination of augmentation methods:* Through the experiment results, we have investigated the best combination of single sampling method and augmentation method on each dataset. We also observe that different methods perform well on the same dataset. For example, "Dimension/Measure + Field Type", "Statistics Feature", "Term Explanation" and "Self Prompting" all show significant improvement on the TabFact dataset. This suggests additional research on the superposable effects through combining multiple augmentation methods. We leave this work for future investigation.

4.4 Comparison Results of Embedding Type.

Based on the results from Table 7, 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-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.

Table 7: 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 [62]	26.76%	62.87%	56.31%	22.65%	47.32%	78.25%
sentence-transformer [51]	26.32%	63.31%	58.94%	23.78%	**50.12%**	80.05%

Table 8: Comparative results of various types of statistical features. The experiment setting is the same as Section 5. 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%**

4.5 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 10. The experiment result, displayed in Table 8, demonstrates that our proposed combination surpasses the previous four feature sets across all six datasets.

4.6 Trade-offs between token allocation

We use five table datasets to find the trade-off between token allocation for table sampling and table augmentation, as demonstrated in Figure 3. We find that:

(1) A balanced token distribution between the table and augmentation (around the 5:5 and 4:6 ratios, also known as a **balanced T:A Ratio**) generally yields the best performance for all five datasets. In indicates that properly controlling the token allocation can help LLMs achieve better performance.

Table 9: Ablation results on five table datasets using gpt-3.5-turbo model. Similar to Table 5, 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.

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%

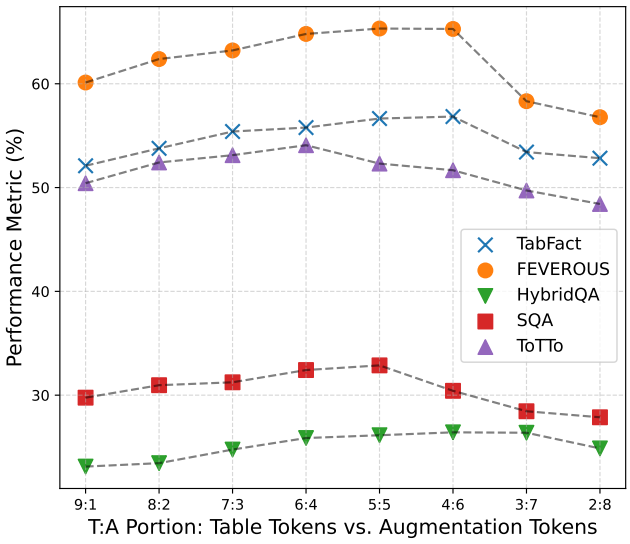

Figure 3: Token Allocation. T:A refers to the ratio of upper #token limitations of sampled table *vs.* augment info.

(2) Diminishing returns are observed when too many tokens are allocated to the augmentation information (as in the 3:7 ratio). This leads to a decrease in performance, suggesting that beyond a certain point, additional augmentation tokens may not be beneficial and could potentially detract from the main table content.

This trade-off highlights a broader concept in data processing and machine learning: the balance between *information overload* and *information scarcity*. Over-augmentation can lead to confusion and difficulty in discerning key patterns or insights. On the other hand, excessive sampling could result in an incomplete or biased understanding of the data. Note that the optimal T: A Ratio may vary across different datasets, as each may have unique characteristics making certain ratios more effective.

4.7 Ablation Study of TAP4LLM

As indicated in Table 9, we conduct an ablation study to assess the impact of various components on the performance of TAP4LLM. Each row represents the method performance without a certain component. We find that each component contributes to the model's effectiveness. The study demonstrates that certain components, such as table sampling and table augmentation, are more crucial. Each dataset reacts differently to the removal of features, which highlights the necessity of a customized design when optimizing for particular datasets.

4.8 Large Table Analysis

Compared to the smaller-sized table, large table can grow to immense sizes, which make it more difficult to efficiently maintain, and reason over the tables. In designing TAP4LLM, performance optimization in this scenario is critical. We plot the distribution regarding the token numbers from the table across the five datasets in Figure 4, and also illustrate the impact of token numbers on LLM performance for three distinct methods: "only sampling", "only augmentation", and the "hybrid method". Note that the "only sampling" method refers to using only the sampling method, without any augmentation methods. Specifically, we use query-based sampling with column grounding for this method. The "only augmentation" method refers to adding only the augmentation information to the prompt, without using any sampling method. In this method, all the augmentation information is simply combined. For the hybrid method, both table sampling and table augmentation methods are used to produce a unified framework. We can observe that:

(1) Data Concentration at Shorter Token Lengths: All datasets show that a majority of the data samples contain a smaller number of tokens, indicated by the frequency distribution that skews towards the left of the graph. This trend suggests that most of the text entries in these datasets are relatively brief.

(2) Performance with Only Augmentation: The method involving only augmentation seems to yield better accuracy at shorter token lengths. This could imply that augmentation techniques are more effective for smaller tables, which may be due to a more focused and less noisy dataset enabling the model to learn and generalize better from less complex input.

(3) Performance with Only Sampling: In contrast, the method that relies only on sampling appears to be more effective for longer token lengths, which correspond to larger tables. This may be because sampling allows the model to handle a greater diversity of data and to process on more informed, longer sequences, thus enhancing its ability to understand and generate correct answers from larger table bodies.

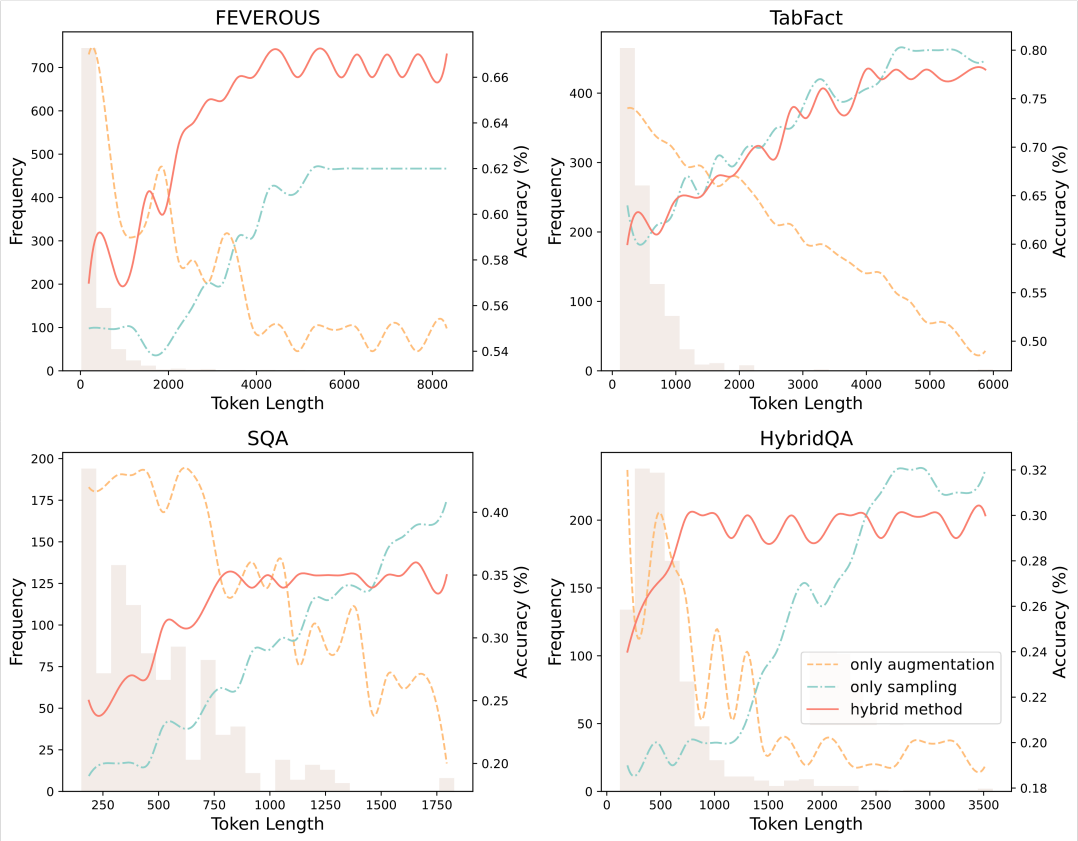

Figure 4: Comparative Analysis of Model Performance Across Four Datasets: FEVEROUS, HybridQA, TabFact, and SQA. The series of graphs illustrates the frequency distribution of token lengths alongside the LLM performance (%) for three distinct methods: only sampling, only augmentation, and the hybrid method. Each subplot corresponds to a different dataset, depicting how table token length impacts model accuracy for various data augmentation and sampling strategies

.

(4) Performance with the Hybrid Method: The hybrid method, which combines both augmentation and sampling techniques, shows a more consistent and stable accuracy across different token lengths. Notably, there's a slight increase in accuracy for larger tables with longer token lengths. This suggests that the hybrid method benefits from the strengths of both augmentation and sampling, providing a balanced approach that ensures stable performance across varying text lengths and complexities.

5 IMPLEMENTATION DETAILS

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.

6 RELATED WORK

Large Language Models Usage Paradigms. With the development of Generative Pre-trained Transformers (GPTs) [6, 45, 46, 49, 50] and other common LLMs [10, 56, 67, 68], many creative ways are proposed: End-to-End, Chain-of-Thought, and Semantic Parsing/-Code Generation. Specifically, End-to-End [7, 23] aims to leverage LLMs to generate final answers directly, often done by providing a task description with a few examples for in-context learning. While convenient to use, this approach has the disadvantage of being uninterpretable and relying solely on the LLMs' training data, which can lead to a lack of robustness and sensitivity to slight input variations. Chain-of-Thought [15, 34, 61] emphasizes breaking down complex

reasoning into a series of intermediate steps. However, this method is unreliable and uncontrollable due to the potential for generated reasoning steps to be misaligned with the intended logical progression, or for incorrect or inconsistent answers to be produced when multiple questions are asked consecutively. Semantic Parsing/Code Generation [11, 14, 64] leverages LLMs to convert natural language queries into executable code or structured representations. This approach enables more precise control over the generated output and facilitates better interpretability. Despite its advantages, this approach has several shortcomings, such as the limited coverage of programming language grammar and semantics. This restricts the model's capacity to handle complex programming tasks and may necessitate the use of additional techniques to effectively process out-of-domain queries. Zhoujun Cheng et al. [11] propose one way to bridge the out-of-domain queries through LLMs. However, how to inject knowledge still lacks a thorough exploration and the trigger functions are still naive for complex queries.

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 [16, 32] and speech [25]. 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 ChatGPT model 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 [24, 36, 37, 71] developed to integrate natural language for tabular data analysis. NL2SQL (Nature language to SQL) [24, 37, 71] is a long-standing research topic that converts natural language to SQL commands that manipulate the relational database. Recently, SheetCopilot [36] explored languages to VBA (Visual Basic for Applications, an embedded script language for Microsoft Excel) command to benefit from a rich set of spreadsheet software functionalities. Compared to previous works, TAP4LLM acts as a dedicated pre-processor designed specifically to improve the efficiency and effectiveness of LLMs for tabular reasoning tasks. We examine the considerable influence of pre-processing on the LLMs' performance when dealing with tabular data. We particularly emphasize the potential of table sampling to extract essential information from the original table, table augmentation to incorporate existing symbolic models and world knowledge, and table packing to manage token allocation for ICL. Our experiment results show that TAP4LLM significantly outperforms the baselines for table reasoning tasks. Beyond its remarkable performance, our system also offers a systematic framework to spur further research into the interpretation of structured data in LLMs.

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 [54, 63], commonsense knowledge [5, 18, 44, 52], and analytical knowledge [19, 30] into the pre-training and inference processes. For example, [30] proposes to semi-automatically transform existing tabular data to create diverse/inventive natural language inference instances for better zero-shot performance. [19] 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. We follow the definition of statistical features from [19]. Each term with the corresponding definition is shown in Table 10. TAP4LLM crowd-source previous table augmentation methods and integrate them for empirical study. This helps us understand which type of knowledge is essential for table reasoning in Language Models (LLMs). We consider this unified knowledge scope to extend the knowledge and reasoning capabilities of LLMs, which enables LLMs to handle a more extensive range of complex tasks with improved accuracy and enhanced contextual understanding.

Table 10: 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

7 CONCLUSION

We present TAP4LLM, a pre-processor designed to enhance the effectiveness of LLMs in tabular reasoning tasks. Technically, our method paves the way for interactive table reasoning as a natural language-driven framework, allowing for different components as plugins. Through three core components: table sampling, table augmentation, and table packing & serialization, we address several major challenges in comprehensive table understanding. We believe that TAP4LLM has the potential to revolutionize table reasoning by enhancing table modeling and exploratory data analysis (EDA) and enabling various domains such as finance, transportation, scientific research,*etc.* Our study will be beneficial for table-based research and serve as an auxiliary tool to help better utilize LLMs on tabular-based/structured data-based tasks.

REFERENCES

[1] [n.d.]. New and Improved Embedding Model. https://openai.com/blog/new-and-improved-embedding-model.

[2] [n.d.]. OpenAI Platform. https://platform.openai.com.

[3] 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. arXiv:2107.06955 [cs] http://arxiv.org/abs/2107.06955

[4] 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)*. Association for Computing Machinery, New York, NY, USA, 2183–2190. https://doi.org/10.1145/3292500.3330693

[5] 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. https://doi.org/10.48550/arXiv.2303.16421 arXiv:2303.16421 [cs]

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[7] Wenhu Chen. 2022. Large Language Models Are Few(1)-Shot Table Reasoners. https://doi.org/10.48550/arXiv.2210.06710 arXiv:2210.06710 [cs]

[8] Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. TabFact: A Large-Scale Dataset for Table-Based Fact Verification. https://doi.org/10.48550/arXiv.1909.02164 arXiv:1909.02164 [cs]

[9] Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020. HybridQA: A Dataset of Multi-Hop Question Answering over Tabular and Textual Data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online, 1026–1036. https://doi.org/10.18653/v1/2020.findings-emnlp.91

[10] 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* (2023).

[11] 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. https://doi.org/10.48550/arXiv.2210.02875 arXiv:2210.02875 [cs]

[12] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. TURL: Table Understanding through Representation Learning. https://doi.org/10.48550/arXiv.2006.14806 arXiv:2006.14806 [cs]

[13] 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)*. Association for Computing Machinery, New York, NY, USA, 317–332. https://doi.org/10.1145/3299869.3314037

[14] Carlos Gemmell and Jeffrey Dalton. 2023. Generate, Transform, Answer: Question Specific Tool Synthesis for Tabular Data. https://doi.org/10.48550/arXiv.2303.10138 arXiv:2303.10138 [cs]

[15] 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*. International Committee on Computational Linguistics, Barcelona, Spain (Online), 1978–1988. https://doi.org/10.18653/v1/2020.coling-main.179

[16] 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* (2023).

[17] 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*. 24–31.

[18] 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. https://doi.org/10.48550/arXiv.2301.07597 arXiv:2301.07597 [cs]

[19] 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*. 9471–9492.

[20] 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*. Association for Computational Linguistics, Online, 4320–4333. https://doi.org/10.18653/v1/2020.acl-main.398

[21] Pascal Hitzler, Aaron Eberhart, Monireh Ebrahimi, Md Kamruzzaman Sarker, and Lu Zhou. 2022. Neuro-Symbolic Approaches in Artificial Intelligence. *National Science Review* 9, 6 (2022), nwac035. https://doi.org/10.1093/nsr/nwac035

[22] Jamie Hoelscher and Amanda Mortimer. 2018. Using Tableau to Visualize Data and Drive Decision-Making. *Journal of Accounting Education* 44 (2018), 49–59. https://doi.org/10.1016/j.jaccedu.2018.05.002

[23] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training Compute-Optimal Large Language Models. arXiv:2203.15556 [cs.CL]

[24] 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* (2023).

[25] 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* (2023).

[26] IDEA-CCNL. 2023. Fengshenbang-LM. https://github.com/IDEA-CCNL/Fengshenbang-LM.

[27] Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. TABBIE: Pretrained Representations of Tabular Data. https://doi.org/10.48550/arXiv.2105.02584 arXiv:2105.02584 [cs]

[28] Baichuan Intelligence. 2023. Baichuan-7B. https://github.com/baichuan-inc/baichuan-7B.

[29] 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)*. Association for Computational Linguistics, Vancouver, Canada, 1821–1831. https://doi.org/10.18653/v1/P17-1167

[30] Aashna Jena, Vivek Gupta, Manish Shrivastava, and Julian Martin Eisenschlos. 2022. Leveraging Data Recasting to Enhance Tabular Reasoning. https://doi.org/10.48550/arXiv.2211.12641 arXiv:2211.12641 [cs]

[31] Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active Retrieval Augmented Generation. arXiv:2305.06983 [cs.CL]

[32] 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* (2023).

[33] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems* 35 (2022), 22199–22213.

[34] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large Language Models Are Zero-Shot Reasoners. https://doi.org/10.48550/arXiv.2205.11916 arXiv:2205.11916 [cs]

[35] LangChain. 2022. LangChain. https://blog.langchain.dev/.

[36] Hongxin Li, Jingran Su, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. 2023. SheetCopilot: Bringing Software Productivity to the Next Level through Large Language Models. *arXiv preprint arXiv:2305.19308* (2023).

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

[38] 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*.

[39] 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. https://doi.org/10.48550/arXiv.2107.07653 arXiv:2107.07653 [cs]

[40] Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, Zihao Wu, Dajiang Zhu, Xiang Li, Ning Qiang, Dingang Shen, Tianming Liu, and Bao Ge. 2023. Summary of ChatGPT/GPT-4 Research and Perspective Towards the Future of Large Language Models. https://doi.org/10.48550/arXiv.2304.01852 arXiv:2304.01852 [cs]

[41] 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*, Vol. 1. Oakland, CA, USA, 281–297.

[42] matthewbolanos. 2023. Orchestrate Your AI with Semantic Kernel. https://learn.microsoft.com/en-us/semantic-kernel/overview/.

[43] 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. arXiv:2112.09332 [cs.CL]

[44] Oluwatosin Ogundare and Gustavo Quiros Araya. 2023. Comparative Analysis of CHATGPT and the Evolution of Language Models. https://doi.org/10.48550/

arXiv.2304.02468 arXiv:2304.02468 [cs]
[45] OpenAI. 2022. ChatGPT. https://openai.com/blog/chatgpt.
[46] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]
[47] 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. https://doi.org/10.48550/arXiv.2203.02155 arXiv:2203.02155 [cs]
[48] Panupong Pasupat and Percy Liang. 2015. Compositional Semantic Parsing on Semi-Structured Tables. arXiv:1508.00305 [cs.CL]
[49] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).
[50] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.
[51] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).
[52] Xinyue Shen, Zeyuan Chen, Michael Backes, and Yang Zhang. 2023. In ChatGPT We Trust? Measuring and Characterizing the Reliability of ChatGPT. https://doi.org/10.48550/arXiv.2304.08979 arXiv:2304.08979 [cs]
[53] 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 (2023).
[54] 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. arXiv:2305.13062 [cs.CL]
[55] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient Transformers: A Survey. https://doi.org/10.48550/arXiv.2009.06732 arXiv:2009.06732 [cs]
[56] 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* (2023).
[57] 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. *Nat Rev Drug Discov* 18, 6 (2019), 463–477. https://doi.org/10.1038/s41573-019-0024-5
[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
[59] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, et al. 2021. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*. 2614–2627.
[60] Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. TUTA: Tree-Based Transformers for Generally Structured Table Pre-Training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1780–1790. https://doi.org/10.1145/3447548.3467434 arXiv:2010.12537 [cs]
[61] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. https://doi.org/10.48550/arXiv.2201.11903 arXiv:2201.11903 [cs]
[62] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-Pack: Packaged Resources To Advance General Chinese Embedding. arXiv:2309.07597 [cs.CL]
[63] 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. https://doi.org/10.48550/arXiv.2201.05966 arXiv:2201.05966 [cs]
[64] Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large Language Models Are Versatile Decomposers: Decompose Evidence and Questions for Table-based Reasoning. https://doi.org/10.48550/arXiv.2301.13808 arXiv:2301.13808 [cs]
[65] 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*. Association for Computational Linguistics, Online, 8413–8426. https://doi.org/10.18653/v1/2020.acl-main.745
[66] 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* (2018).
[67] 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*.
[68] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068* (2022).
[69] Tianping Zhang, Yuanqi Li, Yifei Jin, and Jian Li. 2020. AutoAlpha: An Efficient Hierarchical Evolutionary Algorithm for Mining Alpha Factors in Quantitative Investment. https://doi.org/10.48550/arXiv.2002.08245 arXiv:2002.08245 [q-fin]
[70] 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. arXiv:2303.18223 [cs.CL]
[71] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103* (2017).
[72] 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*. Association for Computational Linguistics, Online, 1393–1404. https://doi.org/10.18653/v1/2021.findings-acl.120