Breaking Size Barrier: Enhancing Reasoning for Large-size Table Question Answering

Xianjie Wu[1], Di Liang[2], Jian Yang[1], Xianfu Cheng[1]
LinZheng Chai[1], Tongliang Li [3] ✉, Liqun Yang[1], Zhoujun Li[1]

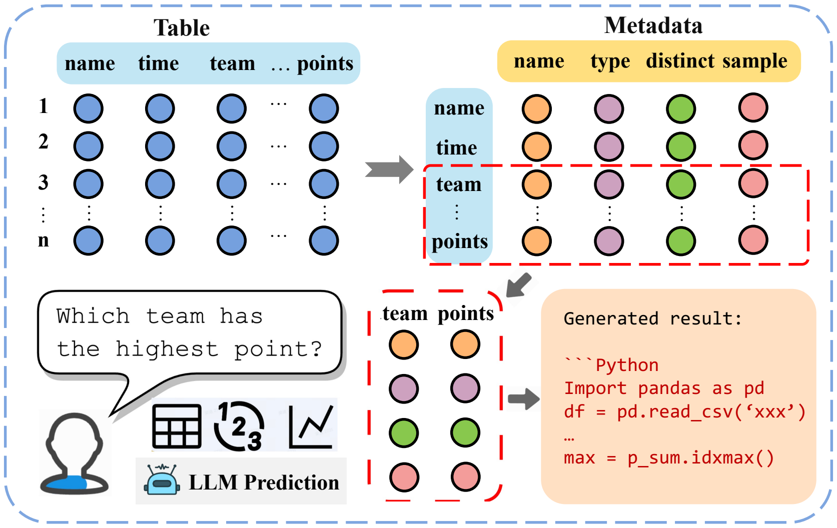
[1]Beihang University, [2]Fudan University, [3]Beijing Info Sci & Tech University
wuxianjie@buaa.edu.cn, tonyliangli@bistu.edu.cn

Abstract. Large language models (LLMs) significantly enhance their ability to process tabular data through chain-of-thought reasoning, particularly in table question answering tasks. However, LLMs encounter substantial challenges when dealing with large tables in real-world applications. Prompting LLMs with the entire table not only encounters context-length constraints but also significantly extends the reasoning path, heightening the risk of reasoning hallucination and information truncation. To address this, we construct a large-size table reasoning (LSTR) benchmark, featuring tables larger than those in existing benchmarks, to thoroughly investigate how table size affects the reasoning abilities of LLMs in answering table-related questions. Subsequently, we propose a size-adaptive-thought (SAT) approach that instructs the LLM utilizing refined metadata to employ Python commands for manipulating tables step by step, thereby facilitating efficient reasoning with tables of any size. Furthermore, we develop `SAT-Llama`, fine-tuned SAT on Llama3.1 (8B), which delivers performance comparable to large-size LLMs at a much lower cost, addressing the issue of inadequate code manipulation capabilities in small-size LLMs. Experimental results on the LSTR and WTQ datasets demonstrate that SAT achieves a new state-of-the-art in handling large-size tables, exhibiting significant performance advantages and high context token efficiency.

Keywords: Table Question Answering · Large Language Model · Large-Size Table

1 Introduction

Large language models (LLMs) revolutionize the field of natural language processing (NLP) by demonstrating exceptional capabilities in understanding and reasoning with complex textual data [12, 21, 1, 38]. Tabular data, characterized by its structured representation of relationships, attributes, and statistical information, serves as a crucial source of knowledge in domains such as financial analysis, scientific research, and business intelligence, where there is a significant demand for table reasoning [39, 37, 29]. Previous works often involve linearizing tables into formats like HTML or JSON to accommodate LLM input [25, 35, 36], while employing step-by-step reasoning [26] or generating executable code [4] to



~~~
Generated result:

```Python
Import pandas as pd
df = pd.read_csv('xxx')
…
max = p_sum.idxmax()
~~~

Fig. 1: An example demonstrates the process of the size-adaptive-thought (SAT).

enhance reasoning capabilities over tabular data. Self-critic methods are introduced to aggregate results from diverse reasoning approaches, enhancing table comprehension [18], while others focus on breaking down complex questions into simpler sub-questions to enable more detailed reasoning [31].

Despite recent advancements, reasoning over large tables presents several challenges. First, LLMs encounter context-length constraints. Recent studies often involve prompting LLMs with entire tables to ensure comprehensive analysis. However, large tables may surpass the context limits of mainstream LLMs, such as the GPT series [3] or LLaMA [9], leading to information truncation and semantic loss. Second, even when table sizes remain within input constraints, long contexts can significantly impair reasoning performance, known as the `"Lost in Middle"` phenomenon [16]. Third, long contexts introduce substantial computational overhead and latency, rendering real-time reasoning applications in industrial settings impractical. *There is a huge need to explore LLMs for reasoning over large-size tables to bridge the gap between real-world applications and academic research.*

Existing TableQA benchmarks, such as WikiTableQuestions (WTQ) [24], Sequential Question Answering (SQA) [11], and TabFact [5], typically involve relatively small tables. In contrast, database benchmarks like DataBench [22], Spider [33], and Bird [14] include larger tables but offer limited diversity. These larger tables often exceed the context-length constraints of most LLMs, thereby failing to adequately address the reasoning challenges posed by large-size tables.

Addressing the benchmark issue, we meticulously construct a large-size table reasoning (LSTR) benchmark, which includes tables significantly larger than those in existing datasets, to tackle reasoning challenges associated with large-sized tables in real-world applications. To break the size barrier in reasoning, we propose a size-adaptive-thought (SAT) approach that enables reasoning over tables of any size with minimal performance compromise. Initially, the SAT method refines table metadata through a rule-based process. Subsequently, the LLM utilizes the metadata, along with the given question, to identify the most pertinent table columns and proceeds to reason through executable Python commands, manipulating the table data step by step to derive the final answer, as illustrated in Figure 1. Moreover, SAT incorporates an error-fixing mechanism that optimizes and debugs the code in response to error messages if execution fails, thereby ensuring the reliability of the generated code. Considering the limitations of small-size LLMs in symbolic code reasoning, we propose the `SAT-Llama`, a fine-tuned SAT approach on Llama3.1 (8B), which achieves comparable performance to large-size LLMs while significantly reducing inference cost and token consumption. Extensive experiments with advanced LLMs across various reasoning in-context learning (ICL) [8] methods on LSTR and WTQ demonstrate the superiority of the SAT method, particularly achieving a new state-of-art in processing large-size tables. Further analysis indicates that with the gradual increase in table size, ICL methods exhibit a noticeable decline in performance and a significant rise in context-token overhead. In contrast, the SAT method demonstrates strong robustness and high efficiency in context-token consumption.

The contributions are summarized as follows:

- We meticulously construct a large-size table reasoning (LSTR) benchmark that includes diverse large tables derived from real-world applications, thereby addressing the limitations of table sizes in existing academic datasets.
- We propose SAT, a size-adaptive-thought (SAT) approach that ensures robustness and efficiency in handling large-size tables. Extensive experiments on LSTR and WTQ validate the superiority of SAT.
- We develop `SAT-Llama`, a fine-tuned open-source LLM that employs the SAT method to achieve performance comparable to larger LLMs while significantly reducing inference costs and token consumption, addressing the shortcomings of small-size LLM in reasoning with large-size tables.

2 Construction of LSTR

Data Collection Due to privacy protection concerns, collecting large-size tables with complex patterns and sufficient value is challenging. Previous works [24, 11, 5, 20, 6] often source table data from Wikipedia; however, the data distribution and patterns in such sources may not accurately reflect real-world scenarios. In our work, we select table data from the Tablib [10] dataset, a compilation of 627 million tables totaling 69 TiB, all of which contain authentic data distributions. Ultimately, We collect 576 tables from the Common Crawl subset of Tablib, covering diverse topics such as finance, competition, sports, and science. To better explore the impact of table size on reasoning tasks in a single dimension, we exclude the impact of complex table structures and content in our table collection process. We choose tables with a simple structure, specifically those with a single-row header, and moderate content difficulty.

Question Annotation Each question is defined as requiring information from a single table for its resolution. We categorize these reasoning questions into two primary types—fact verification and numerical reasoning—and further divide them into nine subtypes, as detailed in Table 2. Five undergraduate students and two Excel experts participate in the annotation process, supported by comprehensive guidelines and an intuitive annotation website. The experts establish standards for annotating both questions and answers, ensuring clarity, precision, and lack of ambiguity in the question statements. They annotate 45 high-quality questions as examples, with five questions for each subcategory, which serve as references for other annotators alongside the established design standards.

Table 1: Data statistics of LSTR

Properties	**Value**
Unique Tables	576
Columns Per Table (Mid/Avg)	7 / 8.02
Rows Per Table (Mid/Avg)	15 / 122.54
Cells Per Table (Mid/Avg)	819 / 4740.77
Question Length (Mid/Avg)	20.30 / 312
Answer Length (Mid/Avg)	8.52 / 31
LSTR Size	688

Table 2: Question Category List

Fact Checking
Match-Based Fact Checking
Multi-hop Fact Checking
Numerical Reasoning
Arithmetic Calculation
Comparison
Aggregation
Ranking
Counting
Time-based Calculation
Multi-hop Numerical Reasoning

Answer Annotation For all answer annotations, we require that answers be expressed in the simplest form, using either a number or an entity name. If multiple answers are present, they should be separated by commas. This format accommodates the validation of results from various reasoning methods. We employ a double-blind mechanism for answer annotation, where two independent annotators generate answers to the same question without discussion. We then assess the consistency of their annotations. In cases of discrepancies, experts review the differing annotations. This review process informs the refinement of our annotation standards, ensuring a comprehensive understanding of the constraints and validity of results for specific question categories.

Dataset Statistic We annotate 688 cases in the LSTR dataset, as illustrated in Table 1, with an average of 197.82 rows, 10.47 columns, 2170.93 cells, and 9508.91 tokens per table, using the JSON format for table representation [25]. Table 3 demonstrates that LSTR is superior is superior to existing benchmarks

Table 3: Comparison with existing advanced datasets

Dataset	Examples	Unique Tables	Rows	Colmuns	Cells	Input Tokens
WTQ [24]	4,344	421	29.53	5.89	171.17	822.5
SQA [11]	3,012	185	14.2	6.09	87.03	407.83
TabFact [5]	12,779	1,695	14.35	6.23	89.18	491.63
FinQA [6]	1,147	379	5.55	3.92	20.56	148.98
TableBench [29]	886	586	16.71	6.68	110.07	546.02
Spider [33]	1,034	80	1229.06	5.11	23769.13	65492.87
Bird [14]	1,534	75	37272.31	10.26	429061.01	4513646.05
DataBench [22]	1,310	65	50307.31	24.82	1155785.61	6472399.53
LSTR	688	**576**	**122.54**	**8.02**	**1073.58**	**4740.77**

in terms of table diversity, reasoning variety, and compatibility with the context-length limits of mainstream LLMs, Addressing the reasoning challenges posed by large-size tables more effectively.

3 Methodology

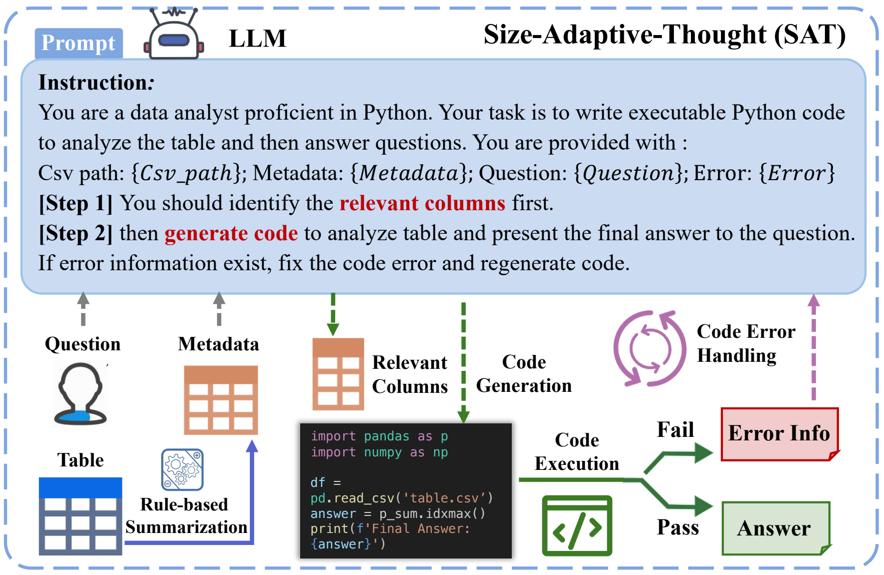
3.1 Task Denfinition

Table question answering (Table QA) can be formulated as follows: Given a semi-structured table $\mathcal{T}$, comprised of $\mathcal{R}$ rows and $\mathcal{C}$ columns, the objective is to generate an answer $\mathcal{A}$ to a question $\mathcal{Q}$ utilizing the information contained within $\mathcal{T}$, where $\mathcal{A}$ is a set of values or entities denoted as $\{a_1, a_2, \ldots, a_k\}$, where $k \in \mathbb{N}^+$.

3.2 Size-Adaptive-Thought (SAT)

To break the length barrier in large-size Table QA tasks, we introduce size-adaptive-thought (SAT), as shown in Figure 2. SAT involves four main stages: (1) constructing the metadata $\mathcal{M}$ of the table $\mathcal{T}$, (2) selecting the most relevant top-k columns $\{c_1, c_2, \ldots, c_k\} \in \mathcal{C}$ to answer the question $\mathcal{Q}$, (3) generating and executing Python code $\mathcal{P}$ for table manipulation, and (4) fixing code errors and regenerating the code until it is executable. For steps 2 to 4, we combine them into a single instruction $\mathcal{I}$ rather than using a multi-agent framework, as each step can serve as a reasoning step to enhance the performance of the subsequent step, which is separated in a multi-agent system.

Meatadata Construction Given the extensive information contained in large tables, we employ table metadata in the SAT to assist the LLMs ($\mathcal{L}$) in comprehending the table structure. The predefined metadata fields are as follows:
- **column-name**: The names assigned to each column.
- **data-type**: The type of data in each column.
- **distinct-enum**: The number of unique values in each column, providing enum insight.



Instruction:
You are a data analyst proficient in Python. Your task is to write executable Python code to analyze the table and then answer questions. You are provided with :
Csv path: {*Csv_path*}; Metadata: {*Metadata*}; Question: {*Question*}; Error: {*Error*}
[Step 1] You should identify the **relevant columns** first.
[Step 2] then **generate code** to analyze table and present the final answer to the question. If error information exist, fix the code error and regenerate code.

```
import pandas as p
import numpy as np

df =
pd.read_csv('table.csv')
answer = p_sum.idxmax()
print(f'Final Answer:
{answer}')
```

Fig. 2: Overview of length-adaptive prompting (SAT).

– **sample-k**: A preview of k entries from each column, providing content insight.
We utilize Python's Pandas library to load table files and construct metadata (denoted as $\mathcal{M}$) based on predefined rules.

Column Selection Even when an original table is condensed into the metadata, the length barrier remains challenging for tables with numerous columns. To address this, we distill the metadata data by selecting the top-k relevant columns. We employ the LLM to identify the columns most relevant to a specific query before code generation, thereby maintaining a consistent and manageable length for query processing, which can be formulated as follows:

$$\mathcal{L}(\mathcal{M},\mathcal{Q}) \rightarrow \mathcal{M}_k = \{c_1,\ldots,c_k\} \subset \mathcal{M} \tag{1}$$

Code Generation and Execution The LLMs take table metadata $\mathcal{M}_k$ and question $\mathcal{Q}$ as inputs, generating executable Python code that utilizes the Pandas library to manipulate the table and derive the answer, formally framed as follows:

$$\mathcal{L}(\mathcal{M}_k,\mathcal{Q}) \rightarrow \mathcal{P} \rightarrow \mathcal{A} \tag{2}$$

Code Error Handling If any errors occur during code execution, SAT collects the error information ($\mathcal{E}$) and incorporates it into the prompt. Based on this information, it optimizes and regenerates the code, repeating the process until executable code is obtained, enhancing the robustness of code construction.

$$\mathcal{L}(\mathcal{M}_k,\mathcal{Q},\mathcal{E}) \rightarrow \mathcal{P} \rightarrow \mathcal{A} \tag{3}$$

3.3 SAT-Llama

Training Set Construction The training of the SAT method does not depend on large tables; instead, we construct the training data by replacing the reasoning processes with SAT in existing benchmarks. We gather training data from widely used Table QA benchmarks, such as WTQ, SQA, and FinQA. By employing GPT-4o, we generate SAT reasoning processes to enhance reasoning capabilities. Executing the code generated by SAT allows us to obtain answers predicted by the LLMs. We then compare these predictions with the ground truth from the benchmark datasets to ensure the validity of the SAT training data. Ultimately, we compile a dataset consisting of 5k SAT training instances.

Supervised Fine-Tuning We fine-tune all parameters of Llama3.1 (8B) [9] using the training set constructed by GPT-4o through the SAT method. The training objective *Loss* for the instruct-tuning is described as:

$$L = -(\sum_i c_i \log(\hat{c}_i) + \sum_t \log P(p_t \mid p_{<t}, \mathcal{T}, \mathcal{Q})) \tag{4}$$

where c_i is the indicator of the selected column, $\hat{c}_i$ is the probability distribution of that column, and $p_t \in \mathcal{P}$ represents the code snippet.

4 Experiments

4.1 Implement Details

We evaluate various in-context learning (ICL) [8] methods and size-adaptive-thought (SAT) on advanced LLMs, including both open-source and proprietary models and supervised fine-tuning (SFT) models [23]. To ensure the accuracy of evaluation results, we impose formatting constraints on the outputs of LLMs and parse the final answers from code execution outputs to eliminate extraneous information. For open-source models, we conduct experiments within the `transformer` environment using multiple A100 GPUs. For proprietary models, we utilize official APIs to interact with exclusive LLMs. During supervised fine-tuning, we employ a cosine annealing scheduler, set the initial learning rate to $2e^{-5}$, and train for three epochs. Optimization is performed using the Adam optimizer with a batch size of 512 and a maximum sequence length of 4096.

4.2 LLMs

We evaluate open-source LLMs on Llama3.1 with 8B and 70B parameters [9]. For proprietary LLMs, we assess the performance of GPT models [3, 21], including GPT-3.5-Turbo and GPT-4-o, as well as Deepseek models [2], specifically Chat-V2. For domain-specific TableQA models, we evaluate TableLlama [35] and TableLLM [36]. Furthermore, we fine-tune `SAT-Llama` based on Llama3.1 (8B) to further explore the Table QA capabilities of LLMs.

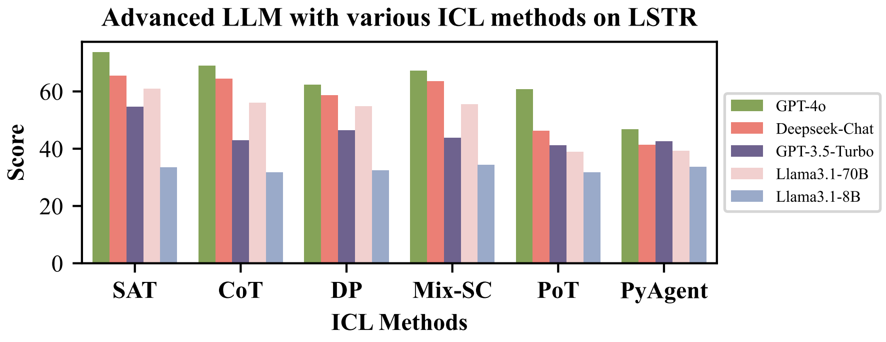


Fig. 3: A bar chart illustrates the performance of advanced LLMs on LSTR.

Table 4: The main results of advanced reasoning ICL methods on LSTR.

	GPT-4o		**Deepseek-Chat**		**GPT-3.5-Turbo**		**Llama3.1-70B**		**Llama3.1-8B**	
	Acc	Rouge-L	Acc	Rouge-L	Acc	Rouge-L	Acc	Rouge-L	Acc	Rouge-L
DP	48.71	62.39	43.81	58.58	31.19	46.47	37.37	54.75	18.56	32.51
CoT	55.41	69.02	53.09	64.44	28.87	43.0	40.72	55.96	20.62	31.76
PyAgent	36.34	46.83	33.33	41.41	29.64	42.58	30.41	39.28	20.10	33.64
Mix-SC	53.61	67.24	53.09	63.58	29.64	43.87	39.95	55.49	21.91	34.34
PoT	48.97	60.69	38.42	46.32	36.86	41.15	32.22	38.82	27.06	31.78
SAT	**60.17**	**73.64**	54.46	65.42	49.43	54.71	49.34	60.94	27.23	33.42

4.3 Baseline Methods

In-context learning (ICL) refers to strategies that optimize input for LLMs (($\mathcal{L}$)) to generate practical outputs with a task-specific instruction ($\mathcal{I}$). We introduce several different ICL methods as baseline methods, as outlined below.

Direct Prompting (DP) [8] introduces a textual ICL method where LLMs respond to queries in a zero-shot format, directly providing the answer without additional context.

Chain of Thought (CoT) [26] proposes a reasoning process where LLMs generate intermediate steps or sub-tasks through textual prompts before arriving at the final answer. These steps collectively form a 'chain of thought' that guides the model towards the correct solution.

Program of Thought (PoT) [4] develops a novel reasoning method that separates computation from reasoning. This approach breaks down the problem into programming commands and uses a language interpreter, such as Python, to compile and execute the generated code.

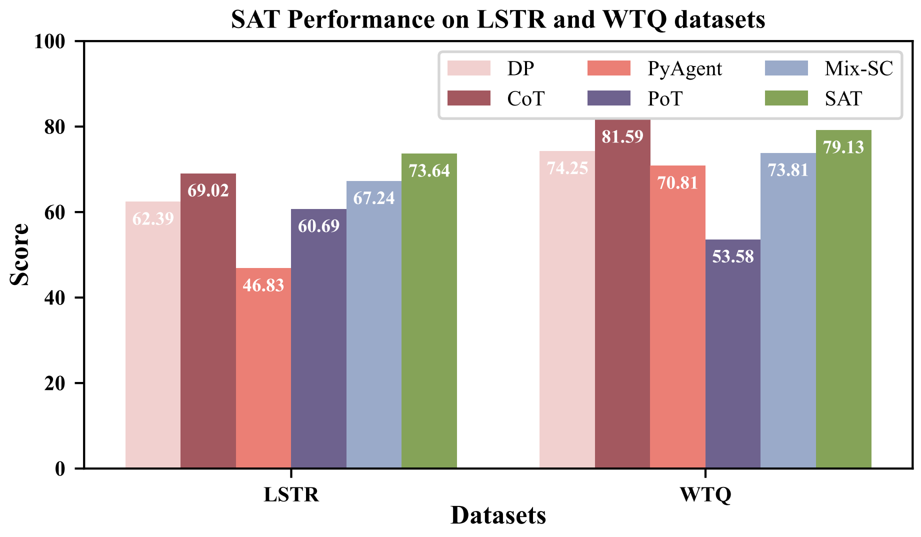


Fig. 4: The results of various ICL methods with GPT4-o on LSTR and WTQ

Python Shell Agent (PyAgent) [32] introduces a symbolic reasoning technique where the model interacts with a Python shell. LLMs use the Python Shell as an external tool to generate commands and analyze results, particularly within a ‘pandas‘ dataframe.

Mix Self-Consistency (Mix-SC) [18] proposes a method that selects a fixed number of outputs for each type of inference (CoT and PyAgent) to achieve self-consistency. This approach is based on the premise that multiple outputs can indicate the confidence levels of LLMs in generating an answer.

4.4 Evaluation Metrics

Automatic Metric We employ Accuracy (Acc) [34] for evaluation, which measures the proportion of questions for generating exactly correct answers. Additionally, we use Rouge-L [15] to assess the quality of the generated answers by measuring the n-gram over SAT with reference answers, increasing the evaluation tolerance for errors due to the free-form generation by LLMs.

LLM-Based Metric Moreover, we introduce an evaluation method, LLM-Eval, instructing GPT-4o [21] to compare the parsed answer with the standard answers, thereby further verifying the consistency of different evaluation metrics.

4.5 Main Results

Table 4 and Figure 3 presents the main results of various advanced LLMs employing different ICL methods on the LSTR. The experimental results indicate

Table 5: `SAT-Llama` performance on LSTR

LLM	**Params**	**Acc**	**Rouge-L**	**LLM-Eval**
Llama3.1-8B	8B	27.23	33.42	29.10
Llama3.1-70B	70B	49.34	60.94	58.51
GPT-3.5-Turbo	proprietary	49.43	54.71	53.09
Deepseek-Chat	proprietary	54.46	65.42	63.66
GPT-4o	proprietary	60.17	73.64	72.86
TableLlama	7B	14.53	24.31	23.93
TableLLM	7B	20.13	29.68	29.14
TableLLM	13B	23.65	31.12	30.56
`SAT-Llama`	8B	**44.64**	**52.19**	**51.96**
Pearson Correlation with Acc		1	0.985	0.990

that SAT significantly outperforms other ICL methods across multiple evaluation metrics, demonstrating a consistent trend across different LLMs. Even with increased table length, SAT maintains stable performance. CoT shows improvements over the DP method, but its effectiveness in managing longer sequences remains limited. Notably, the PoT method, which also relies on code generation and execution, exhibits a performance gap compared to both SAT and CoT. This gap can be attributed to the substantial noise introduced by redundant large-size table inputs. The inherent complexity of code generation tasks makes PoT more susceptible to interference from redundant information compared to CoT. We observe a noticeable decline in performance for both SAT and PoT methods on the smaller Llama3.1-8B model, which is attributed to the model's limited code generation capabilities. Methods related to code generation require not only the ability to perform the task itself (TableQA) but also the capabilities in code generation and instruction adherence.

SAT Performance on WTQ The SAT method still demonstrates an advantage over other context learning methods on the WikiTableQuestions (WTQ) dataset [24], as shown in Figure 4, indicating its strong performance in general table QA tasks. Notably, SAT performs slightly worse than the CoT method on WTQ. Our further case study suggests that this is because tables in WTQ are smaller, allowing the LLM to view the entire table more comprehensively. In such cases, textual-based methods have a clear advantage over code-based operations. Furthermore, by observing the performance of different ICL methods across the two datasets, we find that textual-based ICL methods show significant improvement in WTQ compared to LSTR, particularly the CoT method. The simpler structure and smaller, more intuitive tables in WTQ likely aid the performance of text-based ICL methods, further highlighting the significant challenges that table size poses to LLMs in the LSTR benchmark. Notably, the performance of SAT shows relatively small differences across the two datasets. This obser-

Table 6: Ablation Study on SAT

Method	**Acc**	**Rouge-L**	**LLM-Eval**
$\text{SAT}_{\text{GPT-4o}}$	**60.17**	**73.64**	**72.68**
w/o Metadata	50.48	63.21	62.28
w/o Column Selection	58.67	71.05	71.89
w/o Error Handling	55.15	67.50	67.53

vation highlights the length-adaptive characteristics of SAT, demonstrating its high robustness concerning table size.

SAT-Llama Performance We evaluate the performance of `SAT-Llama` on LSTR, as shown in Table 5. The evaluation results indicate that the SAT-finetuned model `SAT-Llama` significantly reduces the number of model parameters to just 8 billion, yet its performance is comparable to the GPT-3.5 model. This outcome demonstrates the effectiveness of creating cost-efficient models that maintain competitive performance with the SAT method despite their smaller size, highlighting the potential for adopting more efficient and resource-saving solutions without sacrificing accuracy or capability. Additionally, `SAT-Llama` provides valuable insights for practical industrial applications on large tables.

Ablation Study on SAT In this section, we investigate the impact of various modules in SAT, as shown in Table 6. Metadata significantly enhances model performance by efficiently distilling table information, thereby reducing redundancy, especially as table size increases. Consequently, metadata increases the density of useful information, leading to improved performance. In contrast, the improvement provided by column selection is relatively limited, as it only refines information at the column level, offering smaller benefits compared to metadata. The error-handling capability also plays a crucial role in the SAT. Our analysis of experimental data reveals that code error handling with three iterations increases the code pass ratio from 85% to 95%, compared to a single code generation attempt, thereby directly improving model performance.

Consistency of Evaluation Methods Despite imposing restrictions on the format of LLM output and ground-truth annotations, the flexibility of LLM outputs may cause the EM and ROUGE-L metrics to inadequately reflect true performance. Therefore, we also introduce LLM-Eval, an LLM evaluation mechanism based on GPT-4o. We employ the Pearson Correlation Coefficient (PCC) [7] to assess potential biases among different metrics and analyze the consistency between various evaluation methods. As shown in Table 5, the auto-metric and LLM-Eval evaluation methods have a high degree of consistency, indicating that the constraints on LLM outputs are practical and our metrics accurately reflect the real performance of LLMs in benchmarks.

Table 7: Metadata Fields Analysis

Method	Acc	Rouge-L	LLM-Eval
$\text{SAT}_{\text{GPT-4o}}$	**60.17**	**73.64**	**72.68**
w/o colmun name	46.82	59.69	59.23
w/o data type	59.10	72.30	72.47
w/o distinct enum	58.88	72.30	72.11
w/o sample k5	57.23	70.79	70.20

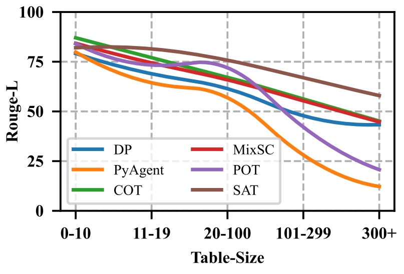
5 Futher Analysis

5.1 Impact of Metadata Field on SAT

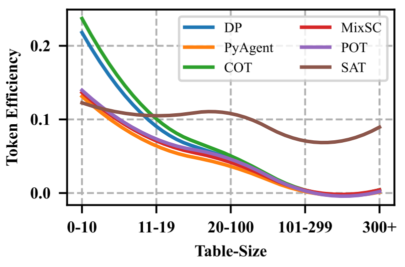
We conducted ablation experiments to investigate the impact of various metadata fields, as illustrated in Table 7. The results indicate that the column name is a crucial factor; without it, the LLM struggles to process the table correctly. The sample k5 significantly optimizes performance by providing example data, which enables the LLM to anticipate the data format in the column and thus better guide its operations. Case studies and the results in the table show that distinct enum categories markedly enhance performance for specific enumerated table types. Listing the enum categories leads to noticeable improvements in these tables. The role of data type is relatively weaker, partly because sample k5 already includes some data type information. However, explicitly specifying the data type still offers some performance benefits. Based on these experimental results, we conclude that using metadata to replace complete table data can simplify and refine the complexity of input data in large-sized tables, aiding the LLM in better understanding the table data.

5.2 Impact of Table Size on ICL Methods

We investigate the impact of table size, measured by the number of rows, on the performance of various ICL methods in the LSTR benchmark. To systematically evaluate this effect, we categorize the number of table rows into five ranges, each containing approximately 60 data points, and calculate the average accuracy within these ranges. Figure 5(a) illustrates the average accuracy of each method across these ranges. It is evident that as the number of cells increases, the performance of ICL methods significantly declines due to the complexity of processing long-context tabular data and the abundance of potential interfering information. Notably, the PoT method exhibits an upward trend in the 20-100 range. Through an in-depth case study, we attribute this to PoT's ability to manipulate tables with coding capacity. When the table size is small, PoT does not show its advantage compared to textual-based methods, which can easily overview the table information and draw direct conclusions. As the table size reaches 20-100, the performance of textual-based methods continues to decline, whereas code-based methods like PoT demonstrate their superiority in manipulating tables



(a) Table size impact analysis



(b) Input token efficiency analysis

Fig. 5: Analysis on table size and token efficiency

through code, resulting in a performance increase that surpasses textual-based methods. However, as the table size further increases, it becomes challenging for LLMs to generate relevant Python commands based on the table content since it is too large to understand, leading to a more pronounced performance decline. Notably, the SAT method, also a code-based approach, exhibits higher robustness as the table size increases. Although there is still a slight downward trend, it is significantly better than other methods, highlighting the superiority of the SAT method for large-size tables.

5.3 Input Token Efficiency of ICL Methods

Since more input tokens imply higher costs, we investigate the input token efficiency across different ICL methods, which is defined as the accuracy divided by the number of input tokens. Similarly, we categorize the number of table rows into five ranges, each containing approximately 60 data points, and calculate the token efficiency within these ranges. As shown in Figure 5, We find that as the size of the table increases, the input token efficiency of ICL methods significantly decreases, indicating that inputting the entire table into the LLM is highly inefficient for large-size tables. Conversely, the SAT method demonstrates relatively stable performance in token efficiency, attributed to its approach of processing large tables into a limited structure of metadata instead of inputting the entire table. However, for smaller tables, ICL methods, particularly CoT and DP, exhibit better token efficiency due to the complex prompt design of SAT and PoT prompting methods, providing insights into the application of different methods for various industrial scenarios.

6 Related Work

Table Question Answering (Table QA) Table QA [19, 13, 29] has advanced significantly due to robust datasets that enhance semantic comprehension. These datasets are pivotal for improving table-centric understanding. WTQ [24], SQA [11], and TabFact [5] provide foundational benchmarks with question-answer pairs from Wikipedia tables. FeTaQA [20] improves model performance with free-form QA datasets requiring answers beyond explicit table content. FinQA [6] focuses on numeric queries in financial tables, presenting complex reasoning challenges. TableLlama [35] offers an open-source corpus and model for table-based tasks, while TableLLM [36] enables LLMs to interact with tabular data, enhancing data handling capabilities. Despite progress, Table QA [25] still needs benchmarks for large table-size issues in real-world scenarios. To address this, LSTR introduces a large-size table reasoning benchmark incorporating real-world challenges, overcoming existing limitations.

Large Language Model (LLM) With rapid advancements in language modeling and pre-training techniques, Transformer-based LLMs like GPT-4o, Claude, PaLM, Qwen2, and LLaMA3 exhibit strong general capabilities [21, 1, 30, 38]. Extensive text pre-training, increased parameter scales, and deeper architectures have endowed LLMs with robust abilities in natural language understanding, generation, generalization, and inference. Integrating LLMs with domain-specific prompts enhances their functionality, particularly in table understanding and table-based question answering, thus improving intention comprehension and complex reasoning. In-context learning (ICL) [28], a recent advancement, allows LLMs to generate task outputs autoregressively based on prompts [17, 27], leveraging pre-trained knowledge and reasoning without additional training.

7 Conclusion

In this work, we address the challenge of reasoning large-size tables in table question answering (Table QA) tasks. To break the size barrier of LLM when processing extensive tables, we propose a large-size table reasoning (LSTR) benchmark, thereby filling a significant gap in the academic research on Table QA for large-size tables. We also introduce the size-adaptive-thought (SAT) approach, which efficiently processes tables of any size while maintaining robustness as table length increases. Moreover, we develop a fine-tuned model, `SAT-Llama`, based on Llama3.1 (8B), which achieves performance comparable to larger models at a significantly lower inference cost. Experiments on the LSTR and WTQ benchmarks demonstrate that SAT offers substantial performance advantages and high token usage efficiency.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 62276017, 62406033, U1636211, 61672081), and the State Key Laboratory of Complex & Critical Software Environment (Grant No. SKLCCSE-2024ZX-18).

References

1. Anil, R., Dai, A.M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., et al.: Palm 2 technical report. arXiv preprint arXiv:2305.10403 (2023)
2. Bi, X., Chen, D., Chen, G., Chen, S., Dai, D., Deng, C., Ding, H., Dong, K., Du, Q., Fu, Z., et al.: Deepseek llm: Scaling open-source language models with longtermism. arXiv preprint arXiv:2401.02954 (2024)
3. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Proc. of NeurIPS pp. 1877–1901 (2020)
4. Chen, W., Ma, X., Wang, X., Cohen, W.W.: Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. TMLR (2022)
5. Chen, W., Wang, H., Chen, J., Zhang, Y., Wang, H., Li, S., Zhou, X., Wang, W.Y.: Tabfact: A large-scale dataset for table-based fact verification. CoRR (2019)
6. Chen, Z., Chen, W., Smiley, C., Shah, S., Borova, I., Langdon, D., Moussa, R., Beane, M., Huang, T.H., Routledge, B.R., et al.: Finqa: A dataset of numerical reasoning over financial data. In: EMNLP 2021. pp. 3697–3711 (2021)
7. Cohen, I., Huang, Y., Chen, J., et al.: Pearson correlation coefficient. Noise reduction in speech processing pp. 1–4 (2009)
8. Dong, Q., Li, L., Dai, D., Zheng, C., Wu, Z., Chang, B., Sun, X., Xu, J., Sui, Z.: A survey for in-context learning. arXiv preprint arXiv:2301.00234 (2022)
9. Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al.: The llama 3 herd of models. arXiv preprint arXiv:2407.21783 (2024)
10. Eggert, G., Huo, K., Biven, M., Waugh, J.: Tablib: A dataset of 627m tables with context. arXiv preprint arXiv:2310.07875 (2023)
11. Iyyer, M., Yih, W.t., Chang, M.W.: Search-based neural structured learning for sequential question answering. In: Proc. of ACL. pp. 1821–1831 (2017)
12. Jiang, A.Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D.S., et al.: Mistral 7b. CoRR (2023)
13. Jin, N., Siebert, J., Li, D., Chen, Q.: A survey on table question answering: recent advances. In: China Conference on Knowledge Graph and Semantic Computing. pp. 174–186 (2022)
14. Li, J., Hui, B., Qu, G., Li, B., Yang, J., Li, B., Wang, B., Qin, B., Cao, R., Geng, R., et al.: Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. arXiv preprint arXiv:2305.03111 (2023)
15. Lin, C.Y.: ROUGE: A package for automatic evaluation of summaries. In: Text Summarization Branches Out. pp. 74–81 (2004)
16. Liu, N.F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., et al.: Lost in the middle: How language models use long contexts. TACL pp. 157–173 (2024)
17. Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G.: Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. ACM Computing Surveys pp. 1–35 (2023)
18. Liu, T., Wang, F., Chen, M.: Rethinking tabular data understanding with large language models. In: Proc. of NAACL. pp. 450–482 (2024)
19. Mueller, T., Piccinno, F., Shaw, P., Nicosia, M., Altun, Y.: Answering conversational questions on structured data without logical forms. In: EMNLP-IJCNLP 2019. pp. 5902–5910 (2019)

20. Nan, L., Hsieh, C., Mao, Z., Lin, X.V., Verma, N., Zhang, R., et al.: Fetaqa: Free-form table question answering. TACL 2022 pp. 35–49 (2022)
21. OpenAI: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
22. Osés Grijalba, J., Ureña-López, L.A., Martínez Cámara, E., Camacho-Collados, J.: Question answering over tabular data with DataBench: A large-scale empirical evaluation of LLMs. In: Proc. of COLING. pp. 13471–13488 (2024)
23. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al.: Training language models to follow instructions with human feedback. Proc. of NeurIPS pp. 27730–27744 (2022)
24. Pasupat, P., Liang, P.: Compositional semantic parsing on semi-structured tables. arXiv preprint arXiv:1508.00305 (2015)
25. Singha, A., Cambronero, J., Gulwani, S., Le, V., Parnin, C.: Tabular representation, noisy operators, and impacts on table structure understanding tasks in llms. arXiv preprint arXiv:2310.10358 (2023)
26. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Proc. of NeurIPS pp. 24824–24837 (2022)
27. Wei, J., Wang, X., Schuurmans, D., Bosma, M., et al.: Chain-of-thought prompting elicits reasoning in large language models. Proc. of NeurIPS pp. 24824–24837 (2022)
28. Wei, J., Wei, J., Tay, Y., Tran, D., Webson, A., Lu, Y., Chen, X., Liu, H., Huang, D., Zhou, D., et al.: Larger language models do in-context learning differently. arXiv preprint arXiv:2303.03846 (2023)
29. Wu, X., Yang, J., Chai, L., Zhang, G., Liu, J., Du, X., Liang, D., Shu, D., Cheng, X., Sun, T., et al.: Tablebench: A comprehensive and complex benchmark for table question answering. arXiv preprint arXiv:2408.09174 (2024)
30. Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., et al.: Qwen2 technical report. arXiv preprint arXiv:2407.10671 (2024)
31. Ye, Y., Hui, B., Yang, M., Li, B., Huang, F., Li, Y.: Large language models are versatile decomposers: Decompose evidence and questions for table-based reasoning. arXiv preprint arXiv:2301.13808 (2023)
32. Ye, Y., Hui, B., Yang, M., Li, B., Huang, F., Li, Y.: Large language models are versatile decomposers: Decompose evidence and questions for table-based reasoning. arXiv preprint arXiv:2301.13808 (2023)
33. Yu, T., Zhang, R., Yang, K., Yasunaga, M., et al.: 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)
34. Zhang, Q., Koudas, N., Srivastava, D., Yu, T.: Aggregate query answering on anonymized tables. In: 2007 IEEE 23rd international conference on data engineering. pp. 116–125 (2006)
35. Zhang, T., Yue, X., Li, Y., Sun, H.: Tablellama: Towards open large generalist models for tables. In: Proc. of NAACL. pp. 6024–6044 (2024)
36. Zhang, X., Zhang, J., Ma, Z., Li, Y., Zhang, B., Li, G., Yao, Z., Xu, K., Zhou, J., Zhang-Li, D., et al.: Tablellm: Enabling tabular data manipulation by llms in real office usage scenarios. arXiv preprint arXiv:2403.19318 (2024)
37. Zhao, B., Ji, C., Zhang, Y., He, W., Wang, Y., et al.: Large language models are complex table parsers. arXiv preprint arXiv:2312.11521 (2023)
38. Zhoujun, L., Yu, F., Xianjie, W.: Review of pre training technology for natural language processing [j]. computer science pp. 162–173 (2020)
39. Zhu, F., Lei, W., Huang, Y., Wang, C., Zhang, S., Lv, J., Feng, F., Chua, T.S.: Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. arXiv preprint arXiv:2105.07624 (2021)