Planning for Success: Exploring LLM Long-term Planning Capabilities in Table Understanding

Thi-Nhung Nguyen[1], **Hoang Ngo**[2], **Dinh Phung**[1], **Thuy-Trang Vu**[1], **Dat Quoc Nguyen**[2]
[1]Monash University, [2]Qualcomm AI Research*
{nhung.thinguyen,dinh.phung,trang.vu1}@monash.edu
{hoangngo,datnq}@qti.qualcomm.com

Abstract

Table understanding is key to addressing challenging downstream tasks such as table-based question answering and fact verification. Recent works have focused on leveraging Chain-of-Thought and question decomposition to solve complex questions requiring multiple operations on tables. However, these methods often suffer from a lack of explicit long-term planning and weak inter-step connections, leading to miss constraints within questions. In this paper, we propose leveraging the long-term planning capabilities of large language models (LLMs) to enhance table understanding. Our approach enables the execution of a long-term plan, where the steps are tightly interconnected and serve the ultimate goal, an aspect that methods based on Chain-of-Thought and question decomposition lack. In addition, our method effectively minimizes the inclusion of unnecessary details in the process of solving the next short-term goals, a limitation of methods based on Chain-of-Thought. Extensive experiments demonstrate that our method outperforms strong baselines and achieves state-of-the-art performance on WikiTableQuestions and TabFact datasets.

1 Introduction

Table understanding is key to addressing challenging downstream tasks involving tables, one of the most prevalent forms of semi-structured data in real-world scenarios, such as table question answering (Wang et al., 2023a; Lin et al., 2023) and fact verification (Chen et al., 2020). The primary goal is to accurately extract relevant information from tables to provide precise answers to user questions. To better understand the problem consider the example in Table 1.

Early works focus on fine-tuning BERT to encode tables (Herzig et al., 2020; Chen et al., 2020). The key idea is to leverage specialized embedding layers or attention mechanisms to encode table cells or segments effectively, enabling models to understand the structure of tables. Another direction revolves around the synthesis of SQL query-response pairs to pre-train an encoder-decoder model as a neural SQL executor (Eisenschlos et al., 2020; Liu et al., 2022b; Jiang et al., 2022). With the advent of large language models (LLMs), recent works have explored instruction fine-tuning of LLMs with tabular data to create generalist models capable of handling a variety of table-based tasks (Zhang et al., 2024), showing improved performance over flagship closed-source LLMs such as `GPT-3.5-turbo` and `GPT-4` (OpenAI et al., 2024).

Product	Region	Sales Year	Quantity Sold	Revenue
A1	A	2023	600	1000
A1	B	2023	400	800
A2	A	2023	700	1500
A2	B	2023	600	1300
A3	A	2022	800	2000
A3	B	2023	300	500

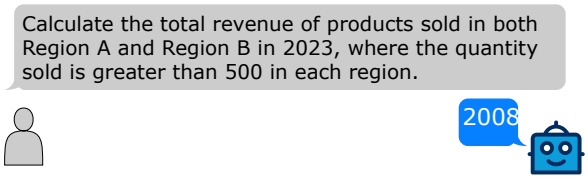

Figure 1: A question-answering example over a table.

Leveraging the strong in-context learning performance of LLMs, recent works have increasingly focused on addressing table understanding through prompting. One common approach is to convert the question into executable languages, allowing the use of tools such as SQL or Python to access the information inside the table (Lin et al., 2023; Gemmell and Dalton, 2023; Wang et al., 2024; Nahid and Rafiei, 2024; Liu et al., 2024; Kong et al., 2024). However, due to the constraints of the single-pass generation process, these methods often struggle with complex questions requiring multiple steps of table operations. To address this

*Qualcomm Vietnam Company Limited.

Proceedings of the 29th Conference on Computational Natural Language Learning, pages 81–92

challenge, some state-of-the-art methods employ Chain-of-Thought (CoT) reasoning, which enables multi-step reasoning (Yao et al., 2023; Chen et al., 2023; Wei et al., 2022; Wang et al., 2024). Others rely on question decomposition, breaking down the question into sub-questions, solving them individually, and finally synthesizing a final answer (Kong et al., 2024; Patnaik et al., 2024; Ye et al., 2023). However, both CoT-based methods and question decomposition-based methods suffer from a lack of explicit long-term planning and weak inter-step connections. This results in missing constraints within the question, leading to incorrect final answers. An illustration of this issue is shown in Figure 2, where step 3 is not conditionally linked to the previous steps. In addition, in the case of CoT-based methods, the entire current chain is often utilized to generate the output for the subsequent step. This approach can result in LLMs forgetting critical details or generating hallucinations, as they process a substantial amount of information, including extraneous details, which may introduce unnecessary complexity and lead to errors (Jiang et al., 2022; Chen, 2023).

In this paper, we propose leveraging the long-term planning capabilities of LLMs to address these challenges. Unlike methods based on CoT and question decomposition, which lack explicit long-term planning, our method begins with the formulation of a long-term plan upon receiving a question. This plan outlines the necessary steps, called short-term goals, to progress systematically from the initial table to the final answer. The short-term goals can be either independent or interconnected, depending on the requirements of the question, ensuring that each serves the long-term goal. To handle each short-term goal effectively, we leverage a set of specialized experts, each dedicated to a specific task. These experts take responsibility for handling short-term goals relevant to their specialization, operating independently to resolve the goals within their localized scope. At this local level, each expert focuses solely on their assigned goal without being influenced by other parts of the long-term plan. The intermediate steps executed by the execution experts are single-pass. Once the short-term goal is completed, only the final results are updated within the long-term plan, minimizing the inclusion of unnecessary information in the process of solving the next short-term goals—a common issue in CoT-based methods.

Our contributions are summarized as follows:

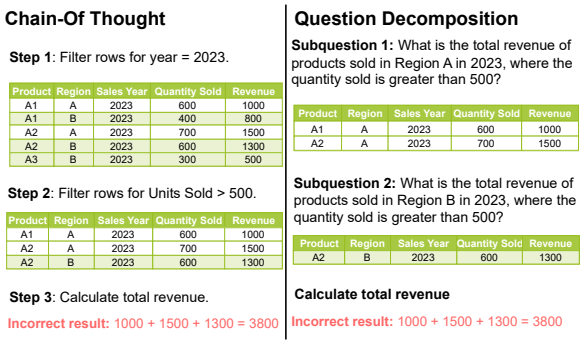

Figure 2: An illustration showing how CoT-based methods and question decomposition-based methods miss the important inter-region condition in revenue calculation (corresponding to the table and question in Figure 1).

(I) We propose leveraging the long-term planning capabilities of LLMs to enhance table understanding. **(II)** Our approach enables the execution of a long-term plan where the steps are tightly interconnected, all serving the ultimate goal—an aspect that methods based on Chain-of-Thought and question decomposition lack. **(III)** Our approach effectively minimizes the inclusion of unnecessary details in the process of solving the next short-term goals—a limitation of methods based on Chain-of-Thought. **(IV)** Comprehensive experiments demonstrate that our approach achieves state-of-the-art performance, outperforming existing strong baselines on standard benchmarks WikiTableQuestions and TabFact.

2 Related Works

Fine-tuning pre-trained BERT models (Devlin et al., 2019) were one the dominant approach for Table Understanding (Herzig et al., 2020; Chen et al., 2020; Liu et al., 2022a; Deng et al., 2022; Wang et al., 2021; Iida et al., 2021). TaPas (Herzig et al., 2020) leverage the mask language modeling approach proposed in BERT to reconstruct certain cells in the table during training process. Wang et al. (2021) further enhance the performance by masking the entire columns in tables. A different approach is to train an encoder-decoder model to transform questions into SQL queries and then answer these questions by executing the respective generated SQL queries (Eisenschlos et al., 2020; Liu et al., 2022b; Jiang et al., 2022). Recently, large language models (LLMs) have demonstrated excellent performance on a variety of tasks. Recent works have been shifting their focus to fine-tuning open-source LLMs to create models capable of handling a variety of table-based tasks. However,

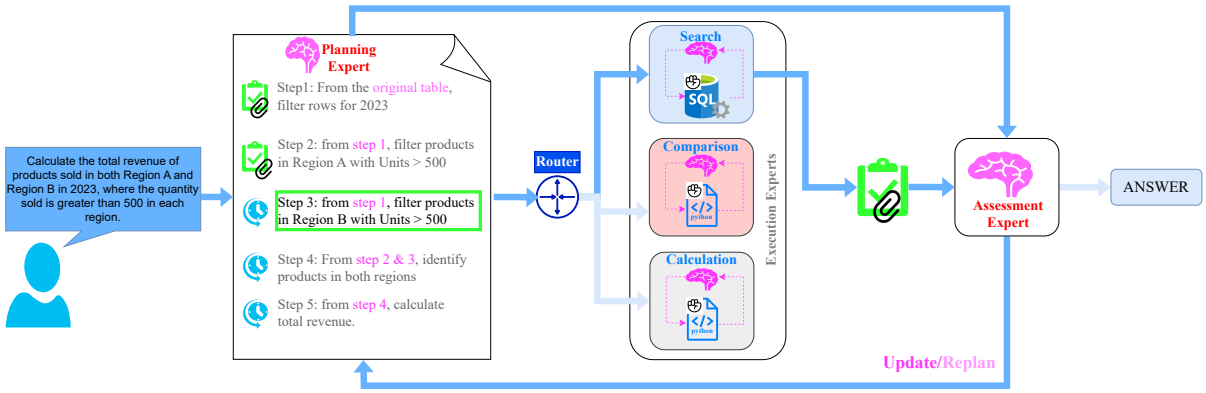

Figure 3: Overview of our proposed framework **PLANTA**.

these methods require expensive labeled data and high training costs. This has led to the emergence of prompt-based approaches, which leverage the in-context learning capabilities of LLMs.

For prompt-based methods, some works propose concatenating task descriptions with the serialized table as a string and inputting them into an LLM to generate a text-based response (Marvin et al., 2023; Cheng et al., 2023; Sui et al., 2024). Other works enhance the performance further by adding few-shot and curated examples to the prompt (Cheng et al., 2023; Narayan et al., 2022; Chen, 2023). However, with reasoning only, LLMs often struggle to accurately retrieve all relevant data required within tables. Therefore, recent works increasingly incorporate external tools (e.g., Python and SQL) instead of relying solely on general text processing to effectively extract relevant data within tables (Chen et al., 2023; Gao et al., 2023; Rajkumar et al., 2022; Cheng et al., 2023; Ni et al., 2023). Despite this, due to the constraint of a single pass, this approach still struggles with complex questions where multiple operations need to be executed to produce an accurate answer. Recent state-of-the-art methods mitigate this limitation by employing chain-of-thought (CoT) reasoning or question decomposition (Chen et al., 2023; Zhao et al., 2024; Yang et al., 2024; Zhou et al., 2023; Khot et al., 2023). Some works (Ye et al., 2023; Cheng et al., 2023; Liu et al., 2024) further enhance the performance by self-consistency technique (Wang et al., 2023b), where a diverse set of reasoning paths is sampled from LLMs and the most consistent answer is selected to obtain the final answer. However, both CoT-based methods and question decomposition-based methods suffer from a lack of explicit long-term planning and weak inter-step connections. This results in constraints within the question being missed, leading to incorrect final answers. Furthermore, CoT-based methods often utilize the entire current chain to generate the output for the subsequent step. This approach can result in LLMs forgetting critical details or generating hallucinations, as they process a substantial amount of information, including extraneous details, which may introduce unnecessary complexity and lead to errors (Jiang et al., 2022; Chen, 2023).

3 Our Approach

We introduce a novel method, named **PLANTA**, which leverages the long-term **Plan**ning capabilities of Large Language Models to improve **Ta**ble Understanding. PLANTA is designed to tackle the challenge of generating accurate answers to table-based questions by extracting and reasoning over relevant information from the given tables.

Figure 3 illustrates the architectural overview of PLANTA. First, upon receiving a table and a question, a Planning expert comes up with a long-term plan outlining the necessary steps, called short-term goals, to transform the initial table into the desired answer to the user's question. Next, each short-term goal is routed to an appropriate Execution expert by a Router, which assigns short-term goals to experts based on their specialization via LLM prompting. These goals are then resolved locally, with only the final results passed to the following components of PLANTA, potentially updating the long-term plan. Meanwhile, intermediate steps executed by the Execution experts are processed in a single pass. After each step, the updated long-term plan is evaluated by an Assessment expert, who determines whether sufficient evidence has been gathered to answer the question or if modifications to the plan are necessary. If no adjustments are needed, the process continues. Below,

Search

```
def execute_sql_query(query, table):
"""
Executes an SQL query
on a table and returns the result.
 Args:
   query: SQL query to execute.
   table: Table to execute the query on.
"""
```

Calculation

```
def multiply(a, b):
  "Returns the product of two numbers."
def minus(a, b):
  "Returns the difference of two numbers"
def sum(a, b):
  "Returns the sum of two numbers"
def divide(a, b):
  "Returns the division of two numbers."
```

Comparison

```
def compare(a, b):
"Return: 'a is greater' if a > b
         'b is greater' if a < b
         'a is equal to b' if a == b"
def max_in_list(a_list):
  "Returns the highest number in the list"
def min_in_list(a_list):
  "Returns the smallest number in the list"
```

Figure 4: Predefined Python functions ("hands") assigned to the Search expert, Calculation expert, and Comparison expert in PLANTA, respectively.

we provide a detailed description of the architecture and roles of the experts within PLANTA. We first outline the common architecture shared by all experts in Subsection 3.1, followed by an in-depth discussion of the differences in their architecture and their specific contributions in Subsection 3.2.

3.1 Common Architecture

In PLANTA, each expert consists of two main components: the "brain" and the "hands". Each *brain* is specialized in a specific task and can independently reason to complete an assigned task. It is powered by an LLM, whose knowledge scope is encoded through prompting. The *hands* are predefined tools, such as Python or SQL execution functions, tailored to the expert's specializations. These tools enable access to detailed data within tables and execute operations that LLMs may struggle with, such as calculations. They provide the brain with the necessary inputs for reasoning and determining the subsequent steps required to complete the task.

3.2 Task-Specific Architecture

Planning expert: Its role is to outline the necessary steps of short-term goals, structured as a task list, to transform the initial table into an accurate answer. Since this role focuses solely on planning without execution, the Planning expert's architecture comprises only the "brain". This brain is powered by an LLM specifically designed for the planning task, with a knowledge scope that includes the given table, the question, and the specializations of Execution experts (see our prompt for Planing in Appendix). For each step in the plan, dependencies on previous steps must be explicitly defined to enable the flexible reuse of variables from earlier steps. This approach minimizes the transfer of unnecessary information to subsequent steps while ensuring that all dependencies are correctly managed. For example, in Figure 3, step 3 depends only on the output of step 1. Therefore, step 3 can access only the output of step 1 that it depends on, without accessing the output of step 2.

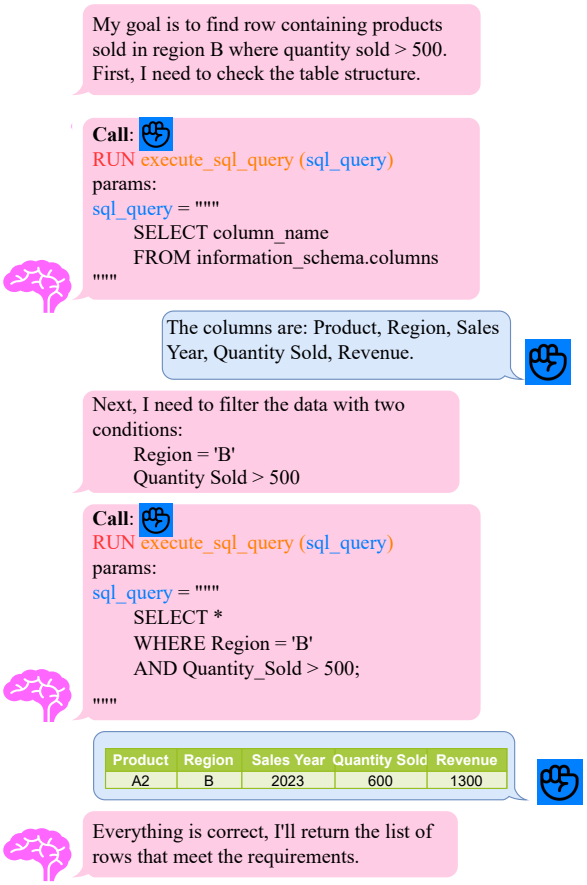

Figure 5: An example of how the Search expert addresses the 3rd short-term goal from Figure 3. Once the short-term goal is received, the Search expert performs reasoning step by step. The process includes understanding the question, analyzing the table structure, identifying the data that meets the goal's conditions, and providing the answer. When the expert needs to access data in the table, it automatically calls the predefined SQL execution function (see Figure 4) to retrieve the necessary information. This is done by generating SQL queries as parameters for the function.

Execution experts: The task of the Execution experts is to resolve the short-term goals required by the Planning expert. These goals are assigned to appropriate Execution experts based on their specialization by the Router, which we perform using an LLM via prompting (see our prompt for the Router in Appendix). Motivated by operations on tables,

our PLANTA system is designed with three Execution experts: (1) Search expert, (2) Comparison expert, and (3) Calculation expert. In terms of their knowledge scope, they can only access the data and short-term goals provided by the Planning expert and reason with the support of predefined functions, which we call "hands". The predefined functions include SQL query execution for the Search expert; comparative and superlative comparison for the Comparison expert; and basic calculations such as addition, subtraction, multiplication, and division for the Calculation expert. See Figure 4 for more details.

Unlike the initial question on tables, which must be addressed according to a pre-defined long-term plan, short-term goals are often simpler and more manageable. Therefore, we allow the experts to independently determine how to resolve assigned goals, such as utilizing the Chain-of-Thought or question decomposition approach, as long as the assigned goal is solved (see our prompts for three Execution experts in Appendix). We expect that this flexibility will enable the experts to reason and use their "hands" in ways that align with their execution capabilities. Figure 5 illustrates an example of how an Execution Expert addresses an assigned short-term goal.

Assessment expert: The task of the Assessment expert is to evaluate the quality of the plan after receiving the result of the current short-term goal from the Execution Expert. Similar to the Planning expert, it does not execute tasks but focuses solely on reasoning, using an LLM specialized in assessment tasks. Its knowledge scope includes access to the table, the question, and the long-term plan (see our prompt for the Assessment in Appendix). After every k short-term goals, the Assessment expert determines whether to generate an early answer if sufficient relevant information is available, or revise the plan if the results from the Execution experts fail to meet requirements or if the initial plan appears infeasible. Otherwise, the results from the Execution experts are automatically incorporated into the long-term plan. In essence, it takes a plan as input and outputs either a revised or an unchanged plan, or an answer.

Recommendation: Our preliminary experiments show that frequent assessments, such as after each short-term goal, can facilitate early answers, conserve resources, and quickly address errors as they arise. However, frequent evaluations may also lead to challenges, such as overemphasizing short-term results at the expense of long-term objectives, unnecessary plan revisions (e.g., repeated short-term result validations), inaccurate premature answers, and increased resource costs. To mitigate these issues, k should be tuned based on the data and the complexity of the question, balancing stability and efficiency.

Statistics	WikiTQ	TabFact
# Questions	4343	2024
# Number of Tables	421	298
# Min/Max Rows	6/518	5/49
# Min/Max Columns	5/20	3/21

Table 1: Statistics of the WikiTableQuestions (WikiTQ) and TabFact test sets.

4 Experiment Setup

Dataset and Metric: Following previous works (Wang et al., 2024), we conduct experiments on the benchmark datasets WikiTableQuestions—a question answering dataset over semi-structured tables (Pasupat and Liang, 2015) and TabFact—a dataset for table-based fact verification (Chen et al., 2020). Table 1 describes the statistics of their test sets. See a description of both datasets in the Appendix.

We employ the official denotation accuracy (Pasupat and Liang, 2015) for WikiTableQuestions and the binary classification accuracy for TabFact.

Baselines: We compare our method to recent strong table understanding methods, including **TEXT2SQL** (Rajkumar et al., 2022), **CHAIN-OF-THOUGHT** (Wei et al., 2022), **Dater** (Khot et al., 2023), **StructGPT** (Jiang et al., 2023), **BINDER** (Cheng et al., 2023), **TabSQLify** (Nahid and Rafiei, 2024), **CHAIN-OF-TABLE** (Wang et al., 2024) and **DP&PYAGENT** (Liu et al., 2024). CHAIN-OF-TABLE and DP&PYAGENT are the state-of-the-art methods on TabFact and WikiTableQuestions, respectively.

Implementation Details: We utilize *LangGraph* to construct our proposed model, PLANTA, which is conceptualized as a graph.[1] In this graph, the long-term plan represents the graph's state and each expert presents a node. Each expert is powered by an LLM with a distinct prompt, as detailed in Appendix . We mainly use "GPT-3.5-turbo" and "GPT-4o-mini" from OpenAI as the LLMs. The *temperature* for LLMs is set to 0. The maximum

[1] https://langchain-ai.github.io/langgraph/

number of iterations for a full turn of reasoning and execution of predefined functions per expert is set to 2. The maximum number of short-term goals is set to 12. The Assessment expert evaluates the long-term plan after completing $n-1$ steps of the plan where n is the number of short-term goals in the plan.

5 Evaluation

5.1 Main Results

Table 2 reports the accuracy of our PLANTA and strong baselines on WikiTableQuestions (WikiTQ) and TabFact test sets.

Recent state-of-the-art methods, including CHAIN-OF-TABLE and DP&PYAGENT, rely on chain-of-thought reasoning and self-consistency, demonstrating the effectiveness of these methods for table understanding. Both CHAIN-OF-TABLE and DP&PYAGENT show notable improvements when upgrading their backbone LLM from GPT-3.5-turbo to GPT-4o-mini. For example, CHAIN-OF-TABLE improves from 59.9 to 70.4 on WikiTQ and 80.2 to 85.8 on TabFact. DP&PYAGENT increases from 65.5 to 74.7 on WikiTQ and 80.0 to 89.9 on TabFact, highlighting the benefits of using a more powerful language model.

Our PLANTA outperforms all baselines on both test sets. With GPT-3.5-turbo, PLANTA scores 70.0 on WikiTQ and 82.0 on TabFact, outperforming DP&PYAGENT (65.5 on WikiTQ, 80.0 on TabFact) and CHAIN-OF-TABLE (59.9 on WikiTQ, 80.2 on TabFact). When using GPT-4o-mini, PLANTA further improves to 75.7 on WikiTQ and 90.4 on TabFact, surpassing DP&PYAGENT (74.7 on WikiTQ, 89.9 on TabFact) and CHAIN-OF-TABLE (70.4 on WikiTQ, 85.8 on TabFact).

Overall, PLANTA demonstrates state-of-the-art performance across different LLMs and datasets, providing clear evidence of the effectiveness of the proposed method for table understanding.

Method	WikiTQ	TabFact
	GPT-3.5-turbo	
TEXT2SQL (2022)	52.9	64.7
CHAIN-OF-THOUGHT	53.5	65.4
BINDER (2023)	56.7	79.2
Dater (2023)	52.8	78.0
StructGPT (2023)	48.4	_
TabSQLify (2024)	64.7	79.5
CHAIN-OF-TABLE (2024)	59.9	80.2
DP&PYAGENT (2024)	65.5	80.0
Our PLANTA	**70.0**	**82.0**
	GPT-4o-mini	
CHAIN-OF-TABLE	70.4	85.8
DP&PYAGENT	74.7	89.9
Our PLANTA	**75.7**	**90.4**

Table 2: Performance results on the WikiTableQuestions (WikiTQ) and TabFact test sets. Rows 3 to 11 evaluate the table understanding capabilities of baseline methods and our PLANTA using GPT-3.5-turbo as the LLM. Results for previous methods are taken from their respective works, except for Dater, BINDER, and DP&PYAGENT. Since original Dater and BINDER relied on the now-decommissioned OpenAI Codex LLM, we extract their results based on GPT-3.5-turbo, reported in the CHAIN-OF-TABLE paper (Wang et al., 2024). Furthermore, DP&PYAGENT is tested only on a variant version of the original WikiTQ test set (i.e. not the same test set). Therefore, we run their official implementation (https://github.com/Leolty/tablellm) to report results on the original WikiTQ and the TabFact test sets with GPT-3.5-turbo. In rows 12-15, we run the official implementations of CHAIN-OF-TABLE (https://github.com/google-research/chain-of-table) and DP&PYAGENT using GPT-4o-mini to provide results with a faster and more cost-efficient LLM. Note that Wang et al. (2024) also report results of CHAIN-OF-TABLE using "PaLM-2" with 340B parameters (Anil et al., 2023). Since the PaLM-2 API has been decommissioned, we are unable to run PLANTA with "PaLM-2".

5.2 Ablation Study

To investigate the impact of each proposed component of PLANTA, we evaluate our ablated variants on WikiTQ and TabFact. Due to budget constraints, we evaluate the ablated variants on **1,000** randomly selected questions from each of the WikiTQ and TabFact test sets. Table 3 presents the contribution of each proposed component to PLANTA's overall performance with GPT-4o-mini.

W/o planning: In this variant, long-term planning is excluded from PLANTA. Instead, the Planning expert relies solely on chain-of-thought (CoT) reasoning. In detail, the Planning expert is required to think step by step and generate a single request for Execution experts to handle. This process is repeated iteratively until a final answer is produced by the Assessment expert. As shown in Table 3, the exclusion of long-term planning significantly hurts PLANTA's performance, with accuracy dropping from 76.5 to 69.0 on WikiTQ and from 90.0 to 74.0 on TabFact. Our internal analysis indicates that the sharper decline on TabFact is due to the

Method	WikiTQ	TabFact
PLANTA$_{\text{GPT-4o-mini}}$	**76.5**	**90.0**
w/o planning	69.0	74.0
w/o search	56.0	62.5
w/o calculation	71.5	81.5
w/o comparison	75.5	88.0
w/o group experts	74.4	88.0
w/o assessment	75.0	85.3

Table 3: The performance of the full-component PLANTA with GPT-4o-mini, along with the results of the ablation study.

nature of fact verification tasks, which typically require only a true/false response. This simplicity may cause the Assessment expert to prematurely decide on an answer without verifying supporting evidence. Meanwhile, WikiTQ questions, which involve more searching tasks, encourage the model to continue processing until the result is found, reducing premature mistakes.

W/o search: In this variant, the Search expert is excluded from PLANTA, and search tasks are instead handled by the Comparison and Calculation experts. This leads to a significant drop in accuracy, from 76.5 to 56.0 on WikiTQ and from 90.0 to 62.5 on TabFact, even though the brains of the Comparison and Calculation experts can still reason to perform searches. These results highlight that search is a critical task, and our design of the Search expert enables the brains to effectively utilize predefined functions, resulting in more accurate search performance compared to relying on reasoning alone.

W/o calculation & W/o comparison: In these variants, the Comparison and Calculation experts are removed from PLANTA separately. Similar to the "W/o search" variant, these exclusions hurt PLANTA's accuracy. Specifically, removing the Calculation expert causes a sharper decline, with a 5% drop on WikiTQ and 8.5% on TabFact, compared to removing the Comparison expert, which results in a 1% drop on WikiTQ and 2% on TabFact, while the Search expert's brain still attempts reasoning to perform these tasks. These results highlight that LLMs' reasoning often struggles with comparison and even basic calculation.

W/o group experts: In this variant, all Execution experts are merged into a single unified expert responsible for handling search, calculation, and comparison tasks. Instead of using specialized prompts and predefined functions tailored to each expert's specific task, the unified expert uses a general prompt and has access to all predefined functions. This consolidation results in a 2.1% drop in accuracy on WikiTQ and a 2.0% drop on TabFact. These results demonstrate that Execution experts benefit significantly from prompts and predefined functions designed specifically for their specialized tasks, highlighting the value of maintaining task-specific experts within PLANTA.

W/o assessment: In this variant, the Assessment expert is excluded from PLANTA. In details, all outputs from the Execution experts are automatically updated into the long-term plan, and the final answer is generated once all short-term goals are completed. Table 3 shows that removing the Assessment expert reduces PLANTA's accuracy by 1.5% on WikiTQ and 4.7% on TabFact. This discrepancy mainly arises from the need to revise the plan to handle code execution errors or situations in which one or more steps in the plan are infeasible, leading to repetitive iterations without returning valid results.

6 Analysis

6.1 Error analysis

Table 4 presents the types of errors observed in PLANTA. The most frequent errors are related to planning and common sense, stemming from the LLMs' lack of *"real expert knowledge"*. As a result, they struggle to handle unpredictable data, such as *"TBA"* for time or *"note" columns* containing additional, contrasting information that alters the main context. This is consistent with the analysis in Subsection 6.2, where we demonstrate that improving the planning capabilities of the LLM leads to a substantial increase in accuracy. The Missing "hands" error, where no predefined function is available to assist reasoning, accounts for only 1.9% of the cases, emphasizing the robustness of our design for predefined functions. However, 11.7% of errors occur when the LLM mistakenly relies solely on reasoning instead of utilizing the available predefined functions to execute tasks accurately. In addition, 11.3% of errors are caused by generating invalid parameters for predefined functions. Hallucinations remain an unavoidable issue with LLMs, accounting for 11.3% of errors. LLMs can generate inaccurate final answers, even when accurate ones are explicitly provided in the final step. Despite its smaller percentage, acceptable answers reflect a need to handle vague questions.

Error Type	Description	%
Planning/Replanning	Errors related to incorrect relationships between steps, failure to handle exceptions within plan, and inability to detect execution errors when revising plan.	37.7%
Common sense	LLMs lack reasoning based on real-world knowledge. Example: When asked how many consecutive years 1990-1991 represent, LLMs answer *"2"*, while the correct answer is 1.	20.8%
Lazy executor	Errors where experts rely solely on LLM reasoning, even when predefined functions could assist, leading to incorrect results. Example: LLMs miscalculate 3 + 3 + 1 = 6, but a tool could compute it correctly.	11.7%
Parameter errors	Errors caused by generating invalid parameters for predefined functions, such as wrong data types or conditions.	11.3%
Hallucination	The plan is executed correctly, but the conclusion is wrong.	11.3%
Acceptable answers	Unclear questions lead to answers that are technically correct but not aligned with the expected response. For example, when asked *row listed before row 4*?, PLANTA lists rows 1 to 3, while the golden answer is row 3.	5.7%.
Missing "hands"	No predefined function is available to support the reasoning process	1.9%

Table 4: Error types in $\text{PLANTA}_{\text{GPT-4o-mini}}$ on the WikiTableQuestions test set. The total percentage does not add up to 100% because some samples contain more than one error.

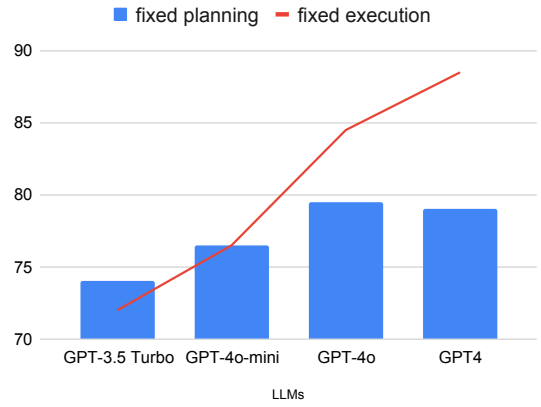

Figure 6: The impact of different LLMs on table understanding performance on the subset of 1000 WikiTableQuestions test questions used for the Ablation study. For "fixed planning", the LLM used for planning is set to GPT-4o-mini, while the LLM for execution tasks varies from GPT-3.5-turbo to GPT-4-mini to GPT-4o to GPT-4, increasing in reasoning capability. For "fixed execution", the LLM for execution tasks remains GPT-4o-mini, with the LLM for planning adjusted from GPT-3.5-turbo to GPT-4-mini to GPT-4o to GPT-4.

6.2 Improved LLMs Are Always the Key to Table Understanding?

As shown in Table 2, using LLMs with better reasoning capabilities notably improves table understanding performance. Here, we investigate whether improvements in LLMs always lead to significant performance gains. Our analysis focuses on two main aspects: [1] **Planning**, which determines how PLANTA chooses the best approach to answer a question, including the task of planning by the Planning expert and the task of revised planning by the Assessment expert; and [2] **Execution**, which involves performing the necessary tasks (such as search, comparison, and calculation) to find the relevant data within tables.

Figure 6 illustrates the impact of different LLMs on table understanding along these two aspects on the subset of 1000 WikiTableQuestions test questions used for the Ablation study. The results demonstrate that under the "fixed execution" setting, planning with better LLMs leads to a substantial improvement in accuracy for table understanding, with GPT-3.5-turbo achieving 72% and GPT-4 increasing this to 88.5%. In contrast, under the "fixed planning" setting, the improvement in execution tasks with better LLMs is far more limited, with accuracy rising from 74% to 79%. This contrast highlights the disproportionate influence of LLM reasoning on planning tasks compared to execution tasks. In other words, execution tasks appear to be less influenced by the model's reasoning power than planning tasks are, emphasizing the need for task-specific optimizations. Thus, by using powerful models for planning and more cost-effective models for execution, we can optimize both performance and resource efficiency.

7 Conclusion

We propose a novel method PLANTA to enhance table understanding by leveraging the long-term planning capabilities of LLMs. Our method focuses on two main goals: (1) enabling the execution of a long-term plan with tightly interconnected steps; (2) minimizing the inclusion of unnecessary details when solving short-term goals, thereby improving efficiency compared to CoT-based approaches. Experimental results show that PLANTA achieves new state-of-the-art performances on two benchmark datasets. Our PLANTA implementation is publicly available at: `https://github.com/nhungnt7/PLANTA`.

Limitations

Although our experiments have proven the effectiveness of our proposed method, there are still some limitations that can be improved in future work. While our approach encourages LLMs to engage in reasoning and solve tasks in a generalist manner, LLMs could benefit significantly from additional task-specific knowledge. For example, providing more targeted few-shot examples and explicitly including common exceptions could help the system handle rare or unpredictable scenarios better, as discussed in Subsection 6.1. Furthermore, future works can impose stricter constraints to encourage LLMs to use the "hands" of predefined functions effectively. This would minimize errors caused by LLMs attempting to rely solely on reasoning when predefined functions are better suited for the task, referred to as lazy executors in Subsection 6.1.

Acknowledgments

This work was supported by Monash eResearch capabilities, including M3.

This work was completed while Hoang Ngo and Dat Quoc Nguyen were at Movian AI, Vietnam.

References

Rohan Anil, Andrew M. Dai, et al. 2023. PaLM 2 Technical Report. *arXiv preprint*, arXiv:2305.10403.

Wenhu Chen. 2023. Large Language Models are few(1)-shot Table Reasoners. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1120–1130.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks. *Transactions on Machine Learning Research*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. TabFact: A Large-scale Dataset for Table-based Fact Verification. In *Proceedings of the 8th International Conference on Learning Representations*.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Binding Language Models in Symbolic Languages. In *Proceedings of the 11th International Conference on Learning Representations*.

Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2022. TURL: Table Understanding through Representation Learning. *ACM SIGMOD Record*, 51(1):33–40.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Julian Eisenschlos, Syrine Krichene, and Thomas Müller. 2020. Understanding tables with intermediate pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 281–296, Online. Association for Computational Linguistics.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided Language Models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 10764–10799.

Carlos Gemmell and Jeff Dalton. 2023. ToolWriter: Question Specific Tool Synthesis for Tabular Data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16137–16148.

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

Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. TABBIE: Pretrained Representations of Tabular Data. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3446–3456.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A General Framework for Large Language Model to Reason over Structured Data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251.

Zhengbao Jiang, Yi Mao, Pengcheng He, Graham Neubig, and Weizhu Chen. 2022. OmniTab: Pretraining with natural and synthetic data for few-shot table-based question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 932–942, Seattle, United States. Association for Computational Linguistics.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed Prompting: A Modular Approach for Solving Complex Tasks. In *Proceedings of the 11th International Conference on Learning Representations*.

Kezhi Kong, Jiani Zhang, Zhengyuan Shen, Balasubramaniam Srinivasan, Chuan Lei, Christos Faloutsos, Huzefa Rangwala, and George Karypis. 2024. OpenTab: Advancing Large Language Models as Open-domain Table Reasoners. In *Proceedings of the 12th International Conference on Learning Representations*.

Weizhe Lin, Rexhina Blloshmi, Bill Byrne, Adria de Gispert, and Gonzalo Iglesias. 2023. An Inner Table Retriever for Robust Table Question Answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9909–9926.

Guang Liu, Jie Yang, and Ledell Wu. 2022a. PTab: Using the Pre-trained Language Model for Modeling Tabular Data. arXiv:2209.08060.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022b. TAPEX: Table pre-training via learning a neural SQL executor. In *International Conference on Learning Representations*.

Tianyang Liu, Fei Wang, and Muhao Chen. 2024. Rethinking Tabular Data Understanding with Large Language Models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 450–482.

Ggaliwango Marvin, Nakayiza Hellen, Daudi Jjingo, and Joyce Nakatumba-Nabende. 2023. Prompt Engineering in Large Language Models. In *Proceedings of the International Conference on Data Intelligence and Cognitive Informatics*, pages 387–402.

Md Nahid and Davood Rafiei. 2024. TabSQLify: Enhancing Reasoning Capabilities of LLMs Through Table Decomposition. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5725–5737.

Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data? *Proceedings of the VLDB Endowment*, 16(4):738–746.

Ansong Ni, Srini Iyer, Dragomir Radev, Ves Stoyanov, Wen-tau Yih, Sida I Wang, and Xi Victoria Lin. 2023. LEVER: learning to verify language-to-code generation with execution. In *Proceedings of the 40th International Conference on Machine Learning*, pages 26106–26128.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, et al. 2024. GPT-4 Technical Report.

Panupong Pasupat and Percy Liang. 2015. Compositional Semantic Parsing on Semi-Structured Tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480.

Sohan Patnaik, Heril Changwal, Milan Aggarwal, Sumit Bhatia, Yaman Kumar, and Balaji Krishnamurthy. 2024. CABINET: Content Relevance-based Noise Reduction for Table Question Answering. In *Proceedings of the 12th International Conference on Learning Representations*.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the Text-to-SQL Capabilities of Large Language Models . *arXiv preprint*, arXiv:2204.00498.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2024. Table Meets LLM: Can Large Language Models Understand Structured Table Data? A Benchmark and Empirical Study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, page 645–654.

Dingzirui Wang, Longxu Dou, and Wanxiang Che. 2023a. A Survey on Table-and-Text HybridQA: Concepts, Methods, Challenges and Future Directions. *arXiv preprint*, arXiv:2212.13465.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *Proceedings of the 11th International Conference on Learning Representations*.

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*, pages 1780–1790.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2024. Chain-of-Table: Evolving Tables in the Reasoning Chain for Table Understanding. In *Proceedings of the 12th International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*.

Bohao Yang, Chen Tang, Kun Zhao, Chenghao Xiao, and Chenghua Lin. 2024. Effective Distillation of Table-based Reasoning Ability from LLMs. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, pages 5538–5550.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large Language Models are Versatile Decomposers: Decomposing Evidence and Questions for Table-based Reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 174–184.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2024. TableLlama: Towards open large generalist models for tables. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6024–6044, Mexico City, Mexico. Association for Computational Linguistics.

Yilun Zhao, Yitao Long, Hongjun Liu, Ryo Kamoi, Linyong Nan, Lyuhao Chen, Yixin Liu, Xiangru Tang, Rui Zhang, and Arman Cohan. 2024. DocMath-eval: Evaluating math reasoning capabilities of LLMs in understanding long and specialized documents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16103–16120.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. In *Proceedings of the 11th International Conference on Learning Representations*.

A Prompt

Table 5 provides details of the custom-designed prompt for each component in the PLANTA system.

B Dataset description

WikiTableQuestions (WikiTQ): A question answering dataset based on HTML tables, each with a minimum of 6 rows and 5 columns. The questions were not generated using predefined templates but were hand-crafted by users, resulting in significant linguistic diversity. These questions span various domains and require operations such as table lookup, aggregation, superlatives, arithmetic operations, joins, and unions.

TabFact: A table-based binary fact verification dataset designed to determine whether a textual hypothesis is supported or refuted based on evidence provided in tables. The dataset presents a challenging task that requires both soft linguistic reasoning and hard symbolic reasoning. TabFact spans a wide range of operations, including aggregation, negation, superlatives, counting, comparative reasoning, and ordinal analysis.

Planning
You are a Planning expert. Your goal is to generate a plan to exclude a sequence of steps including SQL search (more detailed conditions in the requirements are better), calculation, and comparison based on the given table to get the answer to the question. For each step in the plan, dependencies on previous steps must be explicitly defined. Table: *{table}*. Question: *{question}*.

Router
You are a task classification, your task is to classify the requirement type for the given task and route it to the appropriate expert. Please return the expert specialization based on the following guidance: 1. return 'search' if you need to search, conditional count the table for specific information. 2. return 'compare' if you need to compare two or more pieces of information. 3. return 'calculation' if you need to perform a calculation between numbers. Your task: *{short-term goal}*.

Search
You are a Search expert. You have been tasked to reason and generate an SQL query to extract and conditional count specific information (rows) from the table. We allow you to independently determine how to resolve the assigned goal, such as utilizing the Chain-of-Thought or question decomposition approach, as long as the goal is solved. You can use the tool to execute an SQL query generated based on the question and given table and return the result. You might know the answer without running any code, but you should still run the code to get the answer. Given table: *{table}*. Your task: *{shot-term goal}*

Comparision
You are a Comparison expert. You must use the tools provided to complete the assigned task. We allow you to independently determine how to resolve the assigned goal, such as utilizing the Chain-of-Thought or question decomposition approach, as long as the goal is solved. You can use one tool multiple times and use many tools at one time in any order. You might know the answer without running any code, but you should still run the code to get the answer. Your tools include: *{list of predefined functions}*. Your task: *{shot-term goal}*.

Calculation
You are a Calculation expert. You must use the tools provided to complete the assigned task. We allow you to independently determine how to resolve the assigned goal, such as utilizing the Chain-of-Thought or question decomposition approach, as long as the goal is solved. You can use one tool multiple times and use many tools at one time in any order. You might know the answer without running any code, but you should still run the code to get the answer. Your tools include: *{list of predefined functions}*. Your task: *{shot-term goal}* .

Assessment
You are an Assessment expert. Your goal is to answer the question if sufficient relevant information is available or revise the plan if the results from the Execution experts fail to meet requirements or if the initial plan appears infeasible. Your original plan was this: *{plan}*. You have currently done the follow steps with the following results at template (step, result): *{past_steps}*

Table 5: Custom-designed prompts for each component in the PLANTA.