TaPERA: Enhancing Faithfulness and Interpretability in Long-Form Table QA by Content Planning and Execution-based Reasoning

Yilun Zhao[1] **Lyuhao Chen**[2] **Arman Cohan**[1,3] **Chen Zhao**[4]
[1]Yale University [2]Zhejiang University [3]Allen Institute for AI [4]NYU Shanghai

Abstract

Long-form Table Question Answering (LfTQA) requires systems to generate paragraph long and complex answers to questions over tabular data. While Large language models based systems have made significant progress, it often hallucinates, especially when the task involves complex reasoning over tables. To tackle this issue, we propose a new LLM-based framework, TaPERA, for LfTQA tasks. Our framework uses a modular approach that decomposes the whole process into three sub-modules: 1) **QA-based Content Planner** that iteratively decomposes the input question into sub-questions; 2) **Execution-based Table Reasoner** that produces executable Python program for each sub-question; and 3) **Answer Generator** that generates long-form answer grounded on the program output. Human evaluation results on the FeTaQA and QTSumm datasets indicate that our framework significantly improves strong baselines on both accuracy and truthfulness, as our modular framework is better at table reasoning, and the long-form answer is always consistent with the program output. Our modular design further provides transparency as users are able to interact with our framework by manually changing the content plans.

https://github.com/yilunzhao/TaPERA

1 Introduction

Long-form question answering (LFQA) presents a unique challenge in natural language processing (NLP), demanding models that not only comprehend the long context but also maintain factual accuracy during answer generation (Fan et al., 2019; Xu et al., 2023a). A prevalent and concerning issue is *hallucination*, where models generate answers that are coherent yet factually incorrect or irrelevant to the input context. This problem undermines the trustworthiness of NLP systems, posing a barrier to their real-world application in domains where reliability is paramount, such as finance and healthcare. The hallucination issues become even more pronounced when the task involves reasoning over tables, e.g., long-form table question answering (LfTQA) (Nan et al., 2022b; Zhao et al., 2023d).

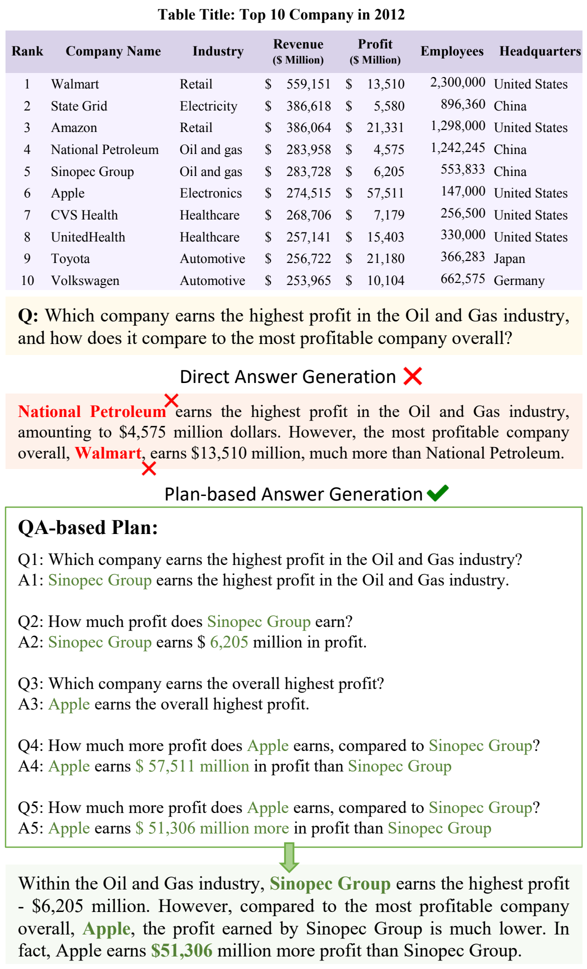


Table Title: Top 10 Company in 2012

Rank	Company Name	Industry	Revenue ($ Million)	Profit ($ Million)	Employees	Headquarters
1	Walmart	Retail	$ 559,151	$ 13,510	2,300,000	United States
2	State Grid	Electricity	$ 386,618	$ 5,580	896,360	China
3	Amazon	Retail	$ 386,064	$ 21,331	1,298,000	United States
4	National Petroleum	Oil and gas	$ 283,958	$ 4,575	1,242,245	China
5	Sinopec Group	Oil and gas	$ 283,728	$ 6,205	553,833	China
6	Apple	Electronics	$ 274,515	$ 57,511	147,000	United States
7	CVS Health	Healthcare	$ 268,706	$ 7,179	256,500	United States
8	UnitedHealth	Healthcare	$ 257,141	$ 15,403	330,000	United States
9	Toyota	Automotive	$ 256,722	$ 21,180	366,283	Japan
10	Volkswagen	Automotive	$ 253,965	$ 10,104	662,575	Germany

Q: Which company earns the highest profit in the Oil and Gas industry, and how does it compare to the most profitable company overall?

Direct Answer Generation ✗

National Petroleum✗ earns the highest profit in the Oil and Gas industry, amounting to $4,575 million dollars. However, the most profitable company overall, **Walmart**,✗ earns $13,510 million, much more than National Petroleum.

Plan-based Answer Generation ✓

QA-based Plan:

Q1: Which company earns the highest profit in the Oil and Gas industry?
A1: Sinopec Group earns the highest profit in the Oil and Gas industry.

Q2: How much profit does Sinopec Group earn?
A2: Sinopec Group earns $ 6,205 million in profit.

Q3: Which company earns the overall highest profit?
A3: Apple earns the overall highest profit.

Q4: How much more profit does Apple earns, compared to Sinopec Group?
A4: Apple earns $ 57,511 million in profit than Sinopec Group

Q5: How much more profit does Apple earns, compared to Sinopec Group?
A5: Apple earns $ 51,306 million more in profit than Sinopec Group

Within the Oil and Gas industry, **Sinopec Group** earns the highest profit - $6,205 million. However, compared to the most profitable company overall, **Apple**, the profit earned by Sinopec Group is much lower. In fact, Apple earns **$51,306** million more profit than Sinopec Group.

Figure 1: An example of long-form table QA. Answering complex questions with an end-to-end LLM-based system poses hallucination challenges (highlighted in Red). We propose a modular-based framework that produces a *QA-based plan* first, followed by generating an answer conditioned on this plan (through generating executable Python programs as an intermediate step).

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 12824–12840
August 11-16, 2024 ©2024 Association for Computational Linguistics

As illustrated in Figure 1, given the complex question and numerous data points in a table, the system must be capable of understanding the relationships within the data and perform human-like reasoning over the tabular content to compose the paragraph-long answer.

This paper introduces TAPERA, aiming to enhance the trustworthiness of LLM-based methods in long-form TAble QA. As depicted in Figure 2, TAPERA employs a modular approach comprising three components: (1) **QA-based Content Planner**: Given a complex question, this module generates and iteratively updates a QA-based content plan. It controls the step-by-step reasoning process to shape the final answer content. (2) **Execution-based Table Reasoner**: For each subquestion in the QA-based plan, this module generates executable programs (i.e. Python programs) to extract and process question-relevant information from tables. (3) **Answer Generator**: Given the sub-question and the executed output from the table reasoner, this module composes a natural language sub-answer, ensuring consistency with the reasoner's output. Moreover, once the entire QA-based plan is finalized, it generates the final answer based on the plan and all sub-answers.

TAPERA features several core advancements: (1) **Modularity**. The complex task is decomposed into multiple easier sub-tasks, which can be solved with more tailored solutions (e.g., reasoning enhanced modules). (2) **Faithfulness** The `table reasoner` first generates an executable Python program as external tools to output question-relevant facts, then the `answer generator` is applied to ensure the consistency between the generated answer and the program output, which improves the faithfulness of generated answer. (3) **Interpretability**. The system allows users for easy identification of which sub-plan or reasoning step leads to an undesired final answer. Additionally, our user study demonstrates the potential of applying user interaction to modify the generated plan to control and improve the final answer.

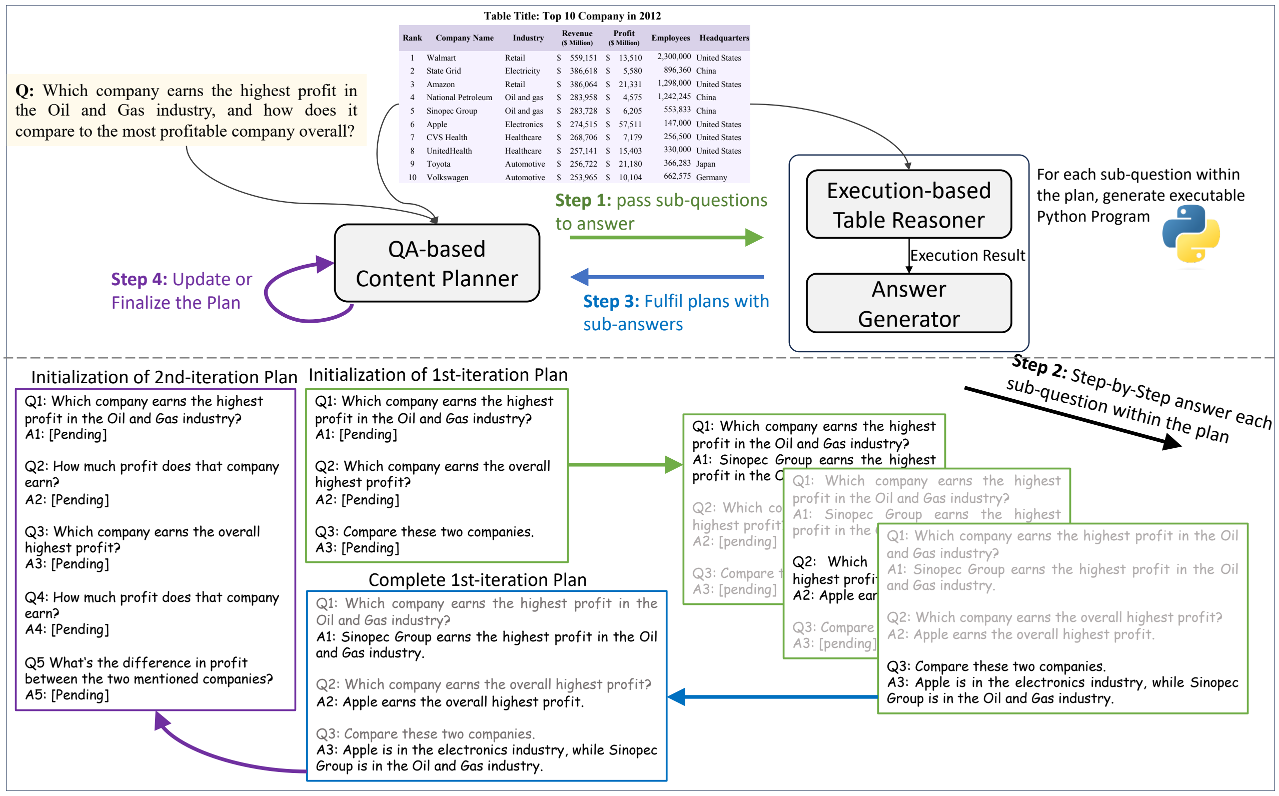
We experiment TAPERA framework on the FETAQA (Nan et al., 2022b) and QTSUMM (Zhao et al., 2023d) datasets. Human evaluation results demonstrate that TAPERA achieves significant improvements in terms of both accuracy and faithfulness. Specifically, TAPERA surpasses 1) the second-best system in comprehensiveness, i.e. Blueprint (Narayan et al., 2023), by 5.4%; and 2) the second-best system in faithfulness, i.e. Dater (Ye et al., 2023), by 5.0%. Our human evaluation results further indicate the inherent limitations in aligning automated metrics and human preference, consistent with previous work (Xu et al., 2023b). In addition, we We also conduct comprehensive human studies, where users are allowed to interact with the system by modifying and improving the QA-based plans generated by the content planner. The results reveal that using such *user-controlled* plans can further improve model performance on those challenging questions.

Our contributions are summarized below:

- We propose TAPERA, a novel modular approach that combines a QA-based content planner and an execution-augmented reasoning module, aiming to enhance the faithfulness and interpretability of LLM-based systems in LFTQA.
- Human evaluation results on the FETAQA and QTSUMM datasets demonstrate that TAPERA achieves significant improvements. Moreover, we conduct detailed analysis on each module.
- Our user study reveals that user interaction with the plans generated by the `QA-based content planner` can further refine and improve model's performance, highlighting the potential of user-driven customization in LLM applications.

2 Related Work

Reasoning over Tabular Data Table reasoning helps non-expert users interact with complex tabular data and has received significant attention. It has several tasks, such as table question answering (Pasupat and Liang, 2015; Iyyer et al., 2017; Zhong et al., 2017; Zhao et al., 2022a; Nan et al., 2022b; Zhao et al., 2023a,b,f), table fact verification (Chen et al., 2020b; Gupta et al., 2020), and table-to-text generation (Chen et al., 2020a; Cheng et al., 2022b; Nan et al., 2022a; Zhao et al., 2023c). Prior work on table reasoning mainly pre-trains models with joint table-text data and then finetunes over specific table-relevant tasks (Herzig et al., 2020; Liu et al., 2022b; Zhao et al., 2022b; Liu et al., 2022a; Jiang et al., 2022; Cheng et al., 2022a; Xie et al., 2022), and recent work shows LLMs also achieves competitive results (Touvron et al., 2023; OpenAI, 2023; Chen, 2023; Zhao et al., 2023e; Tang et al., 2024). To further advance the capabilities of LLMs in table QA tasks, research has primarily focused on two avenues: (1) *Enhancing LLMs with code execution*, which utilize code-optimized LLMs to generate executable programs to answer questions over tabular



Step 1: pass sub-questions to answer
Execution-based Table Reasoner
For each sub-question within the plan, generate executable Python Program
Execution Result
Answer Generator
QA-based Content Planner
Step 4: Update or Finalize the Plan
Step 3: Fulfil plans with sub-answers
Step 2: Step-by-Step answer each sub-question within the plan
Initialization of 2nd-iteration Plan
Initialization of 1st-iteration Plan
Complete 1st-iteration Plan

Q: Which company earns the highest profit in the Oil and Gas industry, and how does it compare to the most profitable company overall?

Table Title: Top 10 Company in 2012

Rank	Company Name	Industry	Revenue ($ Million)	Profit ($ Million)	Employees	Headquarters
1	Walmart	Retail	$ 559,151	$ 13,510	2,300,000	United States
2	State Grid	Electricity	$ 386,618	$ 5,580	896,360	China
3	Amazon	Retail	$ 386,064	$ 21,331	1,298,000	United States
4	National Petroleum	Oil and gas	$ 283,958	$ 4,575	1,242,245	China
5	Sinopec Group	Oil and gas	$ 283,728	$ 6,205	553,833	China
6	Apple	Electronics	$ 274,515	$ 57,511	147,000	United States
7	CVS Health	Healthcare	$ 268,706	$ 7,179	256,500	United States
8	UnitedHealth	Healthcare	$ 257,141	$ 15,403	330,000	United States
9	Toyota	Automotive	$ 256,722	$ 21,180	366,283	Japan
10	Volkswagen	Automotive	$ 253,965	$ 10,104	662,575	Germany

Initialization of 2nd-iteration Plan

Q1: Which company earns the highest profit in the Oil and Gas industry?
A1: [Pending]

Q2: How much profit does that company earn?
A2: [Pending]

Q3: Which company earns the overall highest profit?
A3: [Pending]

Q4: How much profit does that company earn?
A4: [Pending]

Q5 What's the difference in profit between the two mentioned companies?
A5: [Pending]

Initialization of 1st-iteration Plan

Q1: Which company earns the highest profit in the Oil and Gas industry?
A1: [Pending]

Q2: Which company earns the overall highest profit?
A2: [Pending]

Q3: Compare these two companies.
A3: [Pending]

Complete 1st-iteration Plan

Q1: Which company earns the highest profit in the Oil and Gas industry?
A1: Sinopec Group earns the highest profit in the Oil and Gas industry.

Q2: Which company earns the overall highest profit?
A2: Apple earns the overall highest profit.

Q3: Compare these two companies.
A3: Apple is in the electronics industry, while Sinopec Group is in the Oil and Gas industry.

Q1: Which company earns the highest profit in the Oil and Gas industry?
A1: Sinopec Group earns the highest profit in the Oil and Gas industry.

Q2: Which company earns the overall highest profit?
A2: Apple earns the overall highest profit.

Q3: Compare these two companies.
A3: Apple is in the electronics industry, while Sinopec Group is in the Oil and Gas industry.

Figure 2: An illustration of TAPERA. Top: The workflow of our modular framework. The QA-based content planner first generates a plan with QA-pairs, then Step 1-3 fills in the plan with sub-answers, and Step 4 finalizes or refines the plan. Bottom: A running example for the first iteration of the QA-based plan generation and refinement.

data (Cheng et al., 2023; Nan et al., 2023; Kong et al., 2024) or to locate relevant table regions conditioned on a given question (Ye et al., 2023; Patnaik et al., 2024). While this approach is effective for deriving a single fact from tables for short-form answers, it struggles in LFTQA tasks that requires a more complex information synthesis. (2) *Question decomposition*, which breaks down questions into more fine-grained components, thereby simplifying the table reasoning required (Ye et al., 2023; Nan et al., 2023). However, existing methods rely on a single-round question decomposition, which might not always provide sufficient evidence to address the main question in LFTQA.

Content Planning for Text Generation Content planning is effective in capturing and controlling both the content and structure of generated text, particularly for longer outputs (You et al., 2023; Narayan et al., 2023; Huot et al., 2023). The process of plan-based text generation typically involves two distinct stages: planning an outline, and subsequently filling in the specific details. Previous work on table-to-text generation typically represents plans as a sequence of phrase keywords, events, and their interactions (Puduppully et al., 2018; Su et al., 2021; Puduppully and Lapata, 2021; Li et al., 2023), aiming to improve the coherence and narrative flow of generated text. However, such plan representations are not ideally suited for LFTQA tasks, especially in the era of LLMs. This is because the primary objective of LFTQA is to extract and present question-relevant information accurately, rather than creating a narrative flow – an aspect where LLMs already excel (Goyal et al., 2023). This work adapts the idea of QA-based plan (Narayan et al., 2023; Huot et al., 2023; Liu et al., 2023a), utilizing question-answer pairs as intermediate plans for LFTQA tasks. Our work extends beyond previous work on QA-based plans to table-relevant tasks and uses LLMs to iteratively generate and refine the plans based on the feedback (i.e., answer) from table reasoning module.

Automatic Evaluation of LFQA One primary challenge for LFQA lies in accurately evaluating the long-form generation. Recent work (Krishna et al., 2021; Xu et al., 2023b; Krishna et al., 2023; Liu et al., 2023b) indicates that there is no existing automatic text generation metrics are predictive of

Property	FeTaQA	QTSumm
Table Source	Wikipedia	Wikipedia
Unique Tables	1,942	424
Avg. Rows per Table	14.2	12.0
Avg. Columns per Table	5.7	6.7
Avg. Table Title Length	5.4	7.4
Avg. Query Length	14.0	22.3
Avg. Summary Length	23.3	67.8
Test Set Size (# QA Pairs)	2,003	1,078

Table 1: Basic statistics of the FeTaQA and QTSumm test sets used in our experiments.

human preference judgments. We find the same issue when evaluating TaPERA in LfTQA, therefore, we mainly use human evaluation for results.

3 Long-form Table QA Task

In this section, we first formulate the LfTQA task, and then discuss the datasets used in our work.

3.1 Problem Formulation

The LfTQA task can be formulated as follows. The input is a user query $\boldsymbol{Q}$, and a table $\boldsymbol{T}$. The table $\boldsymbol{T} = W \cup \{T_{i,j} | i \leq R_T, j \leq C_T\}$ has R_T rows and C_T columns, with W being the table title and $T_{i,j}$ being the content in the (i, j)-th cell. The task is to generate a paragraph-long answer[1] $\boldsymbol{Y} = (y_1, y_2, \ldots, y_n)$ given the query $\boldsymbol{Q}$ and source table $\boldsymbol{T}$:

$$\boldsymbol{Y} = \text{argmax} \prod_{i=1}^{n} P(y_i | y_{<i}, \boldsymbol{Q}, \boldsymbol{T}; \theta), \quad (1)$$

where θ denotes the parameters of a neural text generation model, and y_i denotes the i-th tokens in the generated answer.

3.2 Evaluated Datasets

We conduct experiments and analysis on the test sets of FeTaQA (Nan et al., 2022b) and QTSumm (Zhao et al., 2023d). Table 1 illustrates the basic data statistics of these two datasets.

- **FeTaQA** is a challenging dataset designed for long-form question answering over tables, with answers averaging 18.9 words. It requires models to fetch multiple entities from the given Wikipedia table, aggregate and reason over these entities, and structure the inferred information to produce a coherent long-form answer.
- **QTSumm** challenges text generation models to perform human-like reasoning and analysis on Wikipedia-sourced tables to produce tailored paragraph-length answers (i.e. summaries). In contrast to the FeTaQA dataset, QTSumm has longer output lengths, with answers averaging 68.0 words.

[1]In the query-focused summarization task (Dang, 2006), the answer $\boldsymbol{Y}$ is also named as summary of the user query.

Algorithm 1: QA-based Plan Generation and Refinement using the TaPERA Framework

```
1:  Input: Complex question $Q$, table $T$
2:  Output: Final answer $A$
3:
4:  $P_0 = \{q_{0,0}, ..., q_{0,n}\}$ ▷ Content planner initializes a plan (Section 4.1)
5:  $i \leftarrow 0$                        ▷ Initialize iteration counter
6:
7:  while $P_i$ is not finalized do            ▷ Iterative Refinement
8:      for Every sub-question $q_{i,j}$ in $P_i$ do
9:          $eo_{i,j}$ ← Table reasoner generates program over $T$ to obtain $q_{i,j}$-relevant execution output (Section 4.2)
10:
11:         $a_{i,j}$ ← Answer generator generates the sub-answer using ($q_{i,j}$, $eo_{i,j}$) (Section 4.3)
12:     end for
13:
14:     $P_i$ ← Content planner updates a new plan based on $P_i$ (Section 4.1)
15:     $i \leftarrow i + 1$
16: end while
17:
18: $A$ ← Answer generator produces a final answer (Section 4.3)
19:
20: return $A$
```

4 Long-form Table QA with TaPERA

In this section, we discuss our proposed framework TaPERA for LfTQA (Figure 2). Solving LfTQA requires complex table reasoning and consistent generation, both are challenging for existing LLM-based systems, and these challenges are more pronounced when they are entangled. Therefore, we propose a modular framework (Andreas et al., 2016) that decomposes the complex task into multiple sub-tasks, and adopt multiple sub-modules to tackle each sub-task. More specifically, our framework contains three main components: a `QA-based content planner`, a `execution-based table reasoner`, and an `answer generator` (Algorithm 1). Given a complex question Q, we generate a QA-based plan that includes a sequence of sub-questions (Section 4.1). Then for each sub-question, we first apply `table reasoner` to generate an executable program to obtain question-relevant facts as program output (Section 4.2). Subsequently, the `answer generator` derives a sub-answer from the program output (Section 4.3). However, due to the complexity of the task, we expect our framework might fail with a single attempt (Narayan et al., 2023). Thus, we adopt an iterative refinement approach that repeats the above process until the `content planner` finalizes a confident QA-based plan. The `answer generator` generates the final answer based on the finalized plan and sub-answers.

4.1 QA-based Content Planner

Previous work on QA-based content generation (You et al., 2023; Narayan et al., 2023; Huot et al., 2023) mainly relies on task-specific models. However, they often struggle to generalize to complex reasoning tasks. We instead adopt LLMs (OpenAI, 2023; Touvron et al., 2023) as a `content planner` for QA-based content plan generation and introduce an iterative refinement pipeline to enhance plan generation. We detail the workflow of `content planner` as follows:

Plan Generation Given a complex question Q over table T, the `content planner` decomposes the question into a series of step-by-step sub-questions. This results into an initial plan P_0 (with empty answers). The finalized plans for the FeTaQA and QTSumm contain an average of 2.6 and 4.7 QA pairs, respectively.

Iterative Plan Refinement The LLMs might not faithfully generate a consistent plan in a single attempt. Additionally, a sub-question might depend on the incorrect answer (from `table reasoner` and `answer generator`) to the previous sub-question, causing cascading errors. Therefore iterative refinement is necessary for consistency and accuracy. Specifically, in each iteration, after getting all sub-answers, we use CoT prompting to instruct LLMs to 1) determine whether the plan is finalized, and 2) refine and generate a revised plan P_{i+1}. If the LLMs determine that the current plan P_i is comprehensive enough to answer the given complex question, we terminate the iteration.

4.2 Execution-Based Table Reasoner

Given a sub-question in the current plan, the LLMs might not generate an accurate answer end-to-end. We therefore propose `execution-based table reasoner` that generates executable programs as an intermediate reasoning step to enhance table reasoning. Unlike recent work in table-relevant tasks (Cheng et al., 2023; Nan et al., 2023; Kong et al., 2024) that uses SQL queries as programs, we use Python programs instead. This is mainly because 1) SQL queries are primarily designed to interact with structured data in relational databases but not most tables in the wild; 2) while SQL is efficient for straightforward data retrieval and basic operations, it becomes less intuitive and user-friendly when dealing with complex table reasoning tasks.

Executable Program Generation Given a sub-question and a table, we prompt `execution-based table reasoner` to generate a Python program. However, in practice, LLMs are likely to generate programs that are not executable. In such cases, we apply Self-Debug prompting methods (Chen et al., 2023), providing LLMs with execution error message to instruct it to regenerate a program.

4.3 Answer Generator

The objective of LLM-based `answer generator` is two-fold: 1) to provide a sub-answer corresponding to the execution output from `table reasoner` in each plan-refinement iteration, and 2) to generate the final answer based on the finalized QA-based plan. The design principle of `answer generator` is not to perform new reasoning over tables, which may introduce hallucinations. Instead, we focus on transforming the output from `table reasoner` into an long-form answer that presents coherent narratives, ensuring the content consistency between the program and the long-form answer.

Generating Sub-Answer from Execution Output We first execute the generated Python program over the table to produce short-form answers, then we prompt `answer generator` to generate a sentence-long answer.

Generating Final Answer Based on Plan Once the QA-based plan is finalized (Section 4.1), the `answer generator` is applied to generate the final answer based on the plan and all generated answers for corresponding sub-questions.

5 Experiment

We next discuss the baselines, human and automated evaluations, implementation details and main experimental results.

5.1 Baseline Systems

We implement the following baseline systems:

- **Zero-/One-shot Direct Answering**. We prompt LLMs to directly output the final answer based on the given table and question.
- **Chain-of-Thought** (Wei et al., 2022) prompts LLMs to generate the reasoning chain in textual format before answering the question.
- **Blueprint** (Narayan et al., 2023) employs a fine-tuned model to generate a QA-based plan for long-form text generation. In our experiment, we adapt the Blueprint method by prompting LLMs to first generate the QA-based plan in a single-round, and then prompt them again to generate the final answer based on the plan.
- **REFACTOR** (Zhao et al., 2023d) employs rule-based methods to extract several question-relevant facts from the table. These facts are then concatenated into the input context for the LLMs.
- **Dater** (Ye et al., 2023) prompts LLMs to break down the table into smaller, question-relevant segments, and then to decompose complex questions into sub-question-answer lists using intermediate SQL queries. Finally, it assembles these elements to compose the final answer.

We also develop three variants of TAPERA to research the effectiveness of each component:

- **TAPERA without Table Reasoner**. In this setting, we remove the component of `execution-based table reasoner` from TAPERA, while keeping the rest the same.
- **TAPERA without Iterative Plan Refinement**. In this setting, `content planner` is only applied to generate the initial plan, with each pending sub-answer fulfilled by `table reasoner` and `answer generator` in each iteration.
- **TAPERA without Sub-Answer Generation**. In this setting, we directly use the execution output from `table reasoner` as the sub-answer.

5.2 Human Evaluation

For human evaluation, we assess the following two dimensions:

- **Faithfulness**: A good answer should correctly answer the given question. It should not contain any unfaithful or hallucinated text.
- **Comprehensiveness**: A good answer should provide all the necessary information to answer the question. Moreover, it should avoid details that are consistent with tabular data yet irrelevant to the given question (Potluri et al., 2023).

Each generated answer was scored from 1 (worst) to 5 (best) for each criteria, with the final score averaged across different evaluators. It is worth noting that we do not evaluate aspects of *fluency* and *coherence*. This is because our preliminary study has found that all the evaluated GPT-based systems are capable of generating coherent and grammatically correct answers. This finding is also corroborated in the QTSUMM paper.

5.3 Automated Evaluation

Following QTSUMM, we adopt following popular automated evaluation metrics:

- **BLEU** (Papineni et al., 2002) computes the geometric average of the precision over output text's n-grams. We use SacreBLEU (Post, 2018) for BLEU score calculation.
- **ROUGE** (Lin and Hovy, 2003) measures the word overlap between the candidate and reference text. We reported F1 score for ROUGE-L.
- **METEOR** (Banerjee and Lavie, 2005) is based on a generalized concept of unigram matching between the generated text and reference.
- **BERTScore** (Zhang et al., 2020) measures the similarity between the reference and generated text using contextual word embeddings.
- **TAPAS-Acc** (Liu et al., 2022a) is a reference-free metric that uses TAPAS (Herzig et al., 2020) fine-tuned on the TabFact dataset (Chen et al., 2020b) as backbone to evaluate the faithfulness of generation. However, since TabFact only contains single-sentence statements, the reliability of using TAPEX-Acc for evaluating long-form generated text remains questionable.
- **AutoACU** (Liu et al., 2023c) is a reference-based evaluation system. The A2CU first extracts atomic content units (ACUs) from the generation and then evaluates them against reference. A3CU is an accelerated version of A2CU that directly computes the similarity between two text without extracting ACUs, but with the similar evaluation target. We use F1 score of A3CU for evaluation.

5.4 Implementation Details

We use `gpt-3.5-turbo-1106` as the backbone for all the evaluated systems. For LLM hyperparameter settings, we set temperature as 0.7, Top P as 1.0, and maximum output length as 512. Following QTSUMM, we represent tabular data in *Markdown* format. In practice, we have found that GPT-*

System	Avg. Answer Length		Comprehensiveness		Faithfulness	
	FeTaQA	QTSumm	FeTaQA	QTSumm	FeTaQA	**QTSumm**
Ground Truth	23.3	67.8				
0-shot	25.6	84.0	3.86	3.70	3.84	3.37
1-shot	29.3	83.0	3.80	3.64	3.91	3.46
CoT	19.4	38.2	3.08	2.96	3.79	3.54
Blueprint	20.2	60.8	3.94	3.72	3.77	3.57
ReFactor	32.9	86.4	3.84	3.54	3.89	3.65
Dater	22.0	43.3	3.75	3.61	3.92	3.76
ours	23.1	52.2	**4.10**	3.99	**4.18**	**4.01**
wo. Iterative Plan Refinement	18.3	44.1	4.06 (-0.04)	**4.04** (+0.05)	3.95 (-0.23)	3.74 (-0.27)
wo. Table Reasoner	21.0	56.2	3.93 (-0.17)	3.88 (-0.11)	4.11 (-0.07)	3.82 (-0.19)
wo. Sub-answer Conversion	20.7	49.7	4.08 (-0.02)	3.91 (-0.08)	4.07 (-0.11)	3.94 (-0.07)

Table 2: Human evaluation results (Likert Scale Scoring) on the aspects of comprehensiveness and faithfulness. Eight human evaluators were engaged to assess 250 predictions from each listed system. An exception was made for TaPERA on QTSumm, where we evaluated all outputs to aid the user analysis discussed in Section 6. We use the faithfulness-level score on QTSumm as the ranking indicator of model performance. All the evaluated systems apply `gpt-3.5-turbo-1106` as the backbone. *: The official implementation does not support the QTSumm dataset; hence, results reported here are from our independent reproduction.

System	BL	R-L	ME	BS	T_{Acc}	ACU
Dater	16.6	35.2	35.5	82.9	83.2	36.3
Blueprint	17.6	38.3	45.7	88.3	86.5	48.1
CoT	19.3	39.0	47.2	85.9	**92.3**	51.9
0-shot Direct	20.1	39.7	**49.7**	89.9	90.5	54.2
ReFactor	19.9	39.5	48.8	**91.2**	86.7	54.7
1-shot Direct	**20.5**	**40.2**	49.5	90.3	89.4	**55.2**
TaPERA	14.6	33.0	33.2	88.7	76.6	37.7
wo. Table Reasoner	14.7	34.4	33.5	88.9	78.8	39.8
wo. Iterative Plan Refine.	14.6	34.5	34.3	88.8	80.4	40.0
wo. Sub-answer Conv.	14.5	34.7	34.2	90.0	78.2	40.4

Table 3: Automated evaluation results on the QTSumm test sets. **BL** denotes BLEU, **R-L** denotes ROUGE-L, **BS** denotes BERTScore, **ME** denotes METEOR, $\mathbf{T}_{Acc}$ denotes TAPAS-Acc. We use **ACU** as the ranking indicator of model performance. The automated evaluation results on FeTaQA are shown in Table 7 in Appendix.

models can effectively understand this table format. To ensure fair comparison, for all modules in the baseline systems and TaPERA, we use a *one-shot* sample if in-context samples are required in the prompts (the first one provided in the official implementation when multiple samples are available). The prompt for each module of TaPERA can be found in the Appendix.

5.5 Main Experimental Results

Table 2 illustrates the results of human evaluation; and Table 3 and Table 7 show the automated evaluation results for QTSumm and FeTaQA, respectively. We draw the following conclusions:

Effectiveness of TaPERA framework Among the evaluated systems, TaPERA achieves the best performance in both faithfulness- and comprehensiveness-level human evaluation, demonstrating the effectiveness of the entire framework. Specifically, TaPERA improves the best baseline in comprehensiveness, Blueprint (which is the worst among baselines in faithfulness) by 5.4%; Similarly, TaPERA improves the best baseline in faithfulness, Dater (one of the worst among baselines in comprehensiveness) by 5.0%. Moreover, the removal of any component from TaPERA leads to a degradation in both comprehensiveness and faithfulness, which underscores the necessity of each module in the system.

Mismatch between automated evaluation and human evaluation Despite receiving the best scores in human evaluation, TaPERA achieves relatively low scores under all automated evaluation metrics. This significant disparity between human evaluation and automated evaluation is consistent with prior work (Xu et al., 2023b), suggesting that future work should focus on developing more reliable automatic evaluation methods.

6 Human Analysis of TaPERA

To obtain a deeper insight into the effectiveness and potential of TaPERA, we conduct a human analy-

Question Type	Representative Question	Description
Open-ended Questions	Summarize the basic information of the episode(s) written by Damon Lindelof	These questions require a comprehensive summary and often have initial plans that lack clarity or specificity.
Subjective Analysis	How did the performance of Tom Brady in terms of passing yards during the Regular Season 2011 compare with other quarterbacks listed in 2011?	These questions involve personal opinions or interpretations, often comparing and contrasting different elements.
Ambiguous Queries	Who were the top three scorers for the 1961-62 Michigan Wolverines men's basketball team and how many points did they score?	Queries that are not clearly defined or have multiple interpretations, leading to vague or inaccurate responses.
Segmentation Errors	What was the overall success of the songs performed in non-English languages in terms of placement and points secured, and were there any notable exceptions to this trend?	These involve errors in dividing or categorizing the question properly, often leading to partial or incorrect analysis.

Table 4: Case studies on four categories of errors from the QA-based content plan generation component. Having human-in-the loop plan generation is a promising direction for improving on these categories.

Error Type	Representative Question	Explanation
Missing or Incorrect Answer in Complex Structure Questions	What are the top three highest mountains in Japan in terms of their elevation in meters and to which prefectures do they belong?	This error occurs due to the model's inability to accurately rank items based on a given table or misunderstand the "top three" requirement, leading to omission or incorrect inclusion.
Misunderstanding the Query	Who were the top three scorers for the 1961-62 Michigan Wolverines men's basketball team and how many points did they score?	The model identifies the top scorers but fails to provide detailed information (individual scores) as requested, indicating a gap in understanding the full scope of the question.
Misunderstanding the Table	How did the team with the lowest conference (Conf.) rank perform in terms of their overall record, Conf. record, PPG, and PAG, and who was their head coach and MVP?	The model incorrectly understands data or key information presented in the table. This typically occurs when the model parses the table and misinterprets numerical values, rankings, or categorical data, leading to incorrect conclusions or predictions.
Inability to Retrieve Relevant Information	Which game had the highest and lowest attendance and what could possibly explain this variance?	This error occurs when the model cannot access or process part of the required information, indicating a limitation in data retrieval or hypothesis generation.
Calculation Error	What is the average duration at the peak for the songs listed in the Adult Contemporary chart from 1961 to 2011?	This error is due to incorrect numerical processing or arithmetic operations by the program, reflecting a flaw in computational accuracy.

Table 5: Case studies on five categories of errors from the reasoning and answer generation modules. Having a better reasoning-enhanced components (or having stronger LLM) can mitigate these errors.

sis on the QTSUMM dataset. Specifically, we collect examples where TAPERA's output received a human evaluation score of 3 or lower in either faithfulness or comprehensiveness (234 out of 1,078 examples). For these erroneous examples, we provide human evaluators with the question, table, and finalized QA-based plan. The evaluators are asked to identify which module (i.e., `content planner`, `table reasoner`, `answer generator`, or a combination thereof) is responsible for the errors. Among the 234 filtered examples, 180 and 77 are marked as errors attributable to the `table reasoner` and `content planner` modules, respectively. Meanwhile, only 12 are marked as errors attributable to `answer generator`, demonstrating that GPT-3.5 performs well in generating answers that are consistent with the execution results or plans. Table 4 and Table 5 present a comprehensive error analysis of `table reasoner` and `content planner`, respectively. In the following subsections, we discuss our analysis of these two modules.

6.1 Analysis of Table Reasoner

The *modularity* feature of TAPERA allows us to update the backbone of each component with more powerful LLMs (i.e., `gpt-4-1106-preview`). We investigate whether using more powerful LLMs as the backbone of `table reasoner` can effectively

Human-Evaluation Criteria	Loss	Tie	**Win**
Faithfulness	15	20	**42**
Comprehensiveness	7	11	**59**

Table 6: Number of losses, ties, and wins for answers generated based on the user-modified plan compared to the original answers. We use the 77 QTSUMM examples marked as errors attributable to the `content planner` for evaluation. The display order of the two answers is randomly shuffled to evaluators during the evaluation.

handle those erroneous examples made by a GPT-3.5. Specifically, for 180 erroneous examples, we repeat the same procedure with the GPT-4-based `table reasoner`, providing it with the same table and sub-questions that led to errors with GPT-3.5. We then manually check the execution results of the GPT-4-based `table reasoner`. We find that *82 out of 180* cases are successfully handled by GPT-4. This significant improvement showcases the vast potential of TAPERA with the continuous advancements in LLMs.

6.2 User Study of Content Planner

The plan-conditioned answer generation offers transparency and interpretability, as it enables users to identify which sub-plan or reasoning step leads to an undesired final answer. In real-world scenarios, it would be beneficial to develop systems that can satisfy users' customized information needs through user-model interaction. Therefore, we conduct a user study where human evaluators are provided with `content planner`-caused erroneous examples. They are then asked to improve the plan by modifying the sub-questions (but not the sub-answers). Then we use the user-modified plan as the initial plan (with all sub-answers set as pending) and apply the same iterative plan-refinement procedure to obtain the answer. This setting ensures that the quality of new answer is based solely on the plan itself, rather than `table reasoner`. We then manually check the quality of the newly generated answers by comparing them with the original ones. The human evaluation results show that user interaction with the content plan can improve answer generation in both comprehensiveness and faithfulness, as illustrated in Table 6.

6.3 Automated Evaluation Case Analysis

To gain a deeper insight into the failure cases of automated evaluation systems for LFTQA, we conducted detailed human analyses by exploring the scenarios where automated evaluations fail. Specifically, we randomly sampled 100 model output pairs from TAPERA and 1-shot Direct on QT-SUMM, where TAPERA received lower scores from at least 4 out of 6 metrics but achieved better results in human evaluations. We chose the 1-shot Direct system for comparison because it demonstrated the best performance in automated evaluations, as shown in Table 3. We meticulously analyzed these failure cases and identified four common scenarios where existing automated metrics tend to fall short: (1) system reveals additional reasoning-intensive information; (2) ground-truth answer is redundant; (3) original question is ambiguous; and (4) ground-truth answer is low-quality and misaligned the question. Detailed explanations for each failure scenarios are provided in Table 8 in Appendix.

7 Conclusion

This work proposes a new modular based framework TAPERA for LFTQA, which decomposes the complex QA task into three sub-tasks: QA-based content plan generation, execution-based table reasoning and consistent answer generation. Human evaluation on two benchmark datasets indicate that TAPERA outperforms strong baselines in both comprehensiveness and faithfulness. This is because TAPERA has stronger table reasoning capacities, and our final answer is consistent with the table reasoning module outputs. We believe this work provides insights for future work in advancing controllable text generation in complex tasks that are often entangled with reasoning challenges.

Limitations and Future Work

There are still some limitations in our work: (1) Due to computational resource constraints, we mainly test our framework on GPT series LLMs. This is because our preliminary analysis indicates that few-shot learning with *open-source LLMs* (e.g., Llama-2 7B) has low performance on LFTQA and struggles to generate executable programs. Future work could explore on fine-tuning these open-source LLMs on LFTQA (Zha et al., 2023; Zhang et al., 2024; Zhuang et al., 2024). (2) Our human study indicates that TAPERA is significantly more faithful over all baselines, and the final answer is always consistent with the program output. However, our `answer generator` module is based on prompting LLMs and cannot avoid hallucination in theory. Replacing `answer generator` with tool-augmented LLMs (Schick et al., 2023) is a promising direction to further tackle hallucination. (3) The findings and conclusions of this paper are mainly based on human evaluation, as none of the current automatic evaluation metrics are predictive of human preference judgments. A possible future direction is to explore fine-grained evaluation metrics that focus on different aspects of generated long-form answers. Moreover, we believe that LLM-based evaluation can also be explored for the LFTQA task (Liu et al., 2023d; Min et al., 2023; Jiang et al., 2024).

Acknowledgement

Chen Zhao is supported by Shanghai Frontiers Science Center of Artificial Intelligence and Deep Learning, NYU Shanghai. We are grateful to the Google TRC program for their support. We would also like to thank the anonymous reviewers and area chairs for constructive feedback.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 39–48.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Wenhu Chen. 2023. Large language models are few(1)-shot table reasoners. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1120–1130, Dubrovnik, Croatia. Association for Computational Linguistics.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020a. Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942, Online. Association for Computational Linguistics.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020b. Tabfact: A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations*.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug.

Zhoujun Cheng, Haoyu Dong, Ran Jia, Pengfei Wu, Shi Han, Fan Cheng, and Dongmei Zhang. 2022a. FORTAP: Using formulas for numerical-reasoning-aware table pretraining. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1150–1166, Dublin, Ireland. Association for Computational Linguistics.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2022b. HiTab: A hierarchical table dataset for question answering and natural language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1094–1110, Dublin, Ireland. Association for Computational Linguistics.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Binding language models in symbolic languages. In *The Eleventh International Conference on Learning Representations*.

Hoa Trang Dang. 2006. DUC 2005: Evaluation of question-focused summarization systems. In *Proceedings of the Workshop on Task-Focused Summarization and Question Answering*, pages 48–55, Sydney, Australia. Association for Computational Linguistics.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. Eli5: Long form question answering. *arXiv preprint arXiv:1907.09190*.

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2023. News summarization and evaluation in the era of gpt-3.

Vivek Gupta, Maitrey Mehta, Pegah Nokhiz, and Vivek Srikumar. 2020. INFOTABS: Inference on tables as semi-structured data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2309–2324, Online. Association for Computational Linguistics.

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

Fantine Huot, Joshua Maynez, Shashi Narayan, Reinald Kim Amplayo, Kuzman Ganchev, Annie Priyadarshini Louis, Anders Sandholm, Dipanjan Das, and Mirella Lapata. 2023. Text-blueprint: An interactive platform for plan-based conditional generation. In *Proceedings of the 17th Conference of* *the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 105–116, Dubrovnik, Croatia. Association for Computational Linguistics.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831, Vancouver, Canada. Association for Computational Linguistics.

Dongfu Jiang, Yishan Li, Ge Zhang, Wenhao Huang, Bill Yuchen Lin, and Wenhu Chen. 2024. Tigerscore: Towards building explainable metric for all text generation tasks.

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.

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 *The Twelfth International Conference on Learning Representations*.

Kalpesh Krishna, Erin Bransom, Bailey Kuehl, Mohit Iyyer, Pradeep Dasigi, Arman Cohan, and Kyle Lo. 2023. Longeval: Guidelines for human evaluation of faithfulness in long-form summarization. *arXiv preprint arXiv:2301.13298*.

Kalpesh Krishna, Aurko Roy, and Mohit Iyyer. 2021. Hurdles to progress in long-form question answering. *arXiv preprint arXiv:2103.06332*.

Liang Li, Ruiying Geng, Chengyang Fang, Bing Li, Can Ma, Binhua Li, and Yongbin Li. 2023. Plan-then-seam: Towards efficient table-to-text generation. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 205–219, Dubrovnik, Croatia. Association for Computational Linguistics.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 150–157.

Ao Liu, Haoyu Dong, Naoaki Okazaki, Shi Han, and Dongmei Zhang. 2022a. PLOG: Table-to-logic pretraining for logical table-to-text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5531–5546, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Danyang Liu, Mirella Lapata, and Frank Keller. 2023a. Visual storytelling with question-answer plans. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5800–5813, Singapore. Association for Computational Linguistics.

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*.

Yixin Liu, Alex Fabbri, Pengfei Liu, Yilun Zhao, Linyong Nan, Ruilin Han, Simeng Han, Shafiq Joty, Chien-Sheng Wu, Caiming Xiong, and Dragomir Radev. 2023b. Revisiting the gold standard: Grounding summarization evaluation with robust human evaluation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4140–4170, Toronto, Canada. Association for Computational Linguistics.

Yixin Liu, Alexander Fabbri, Yilun Zhao, Pengfei Liu, Shafiq Joty, Chien-Sheng Wu, Caiming Xiong, and Dragomir Radev. 2023c. Towards interpretable and efficient automatic reference-based summarization evaluation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16360–16368, Singapore. Association for Computational Linguistics.

Yixin Liu, Alexander R. Fabbri, Jiawen Chen, Yilun Zhao, Simeng Han, Shafiq Joty, Pengfei Liu, Dragomir Radev, Chien-Sheng Wu, and Arman Cohan. 2023d. Benchmarking generation and evaluation capabilities of large language models for instruction controllable summarization.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

Linyong Nan, Lorenzo Jaime Flores, Yilun Zhao, Yixin Liu, Luke Benson, Weijin Zou, and Dragomir Radev. 2022a. R2D2: Robust data-to-text with replacement detection. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6903–6917, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, Mutethia Mutuma, Ben Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, Dragomir Radev, and Dragomir Radev. 2022b. FeTaQA: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

Linyong Nan, Ellen Zhang, Weijin Zou, Yilun Zhao, Wenfei Zhou, and Arman Cohan. 2023. On evaluating the integration of reasoning and action in llm agents with database question answering.

Shashi Narayan, Joshua Maynez, Reinald Kim Amplayo, Kuzman Ganchev, Annie Louis, Fantine Huot, Anders Sandholm, Dipanjan Das, and Mirella Lapata. 2023. Conditional generation with a question-answering blueprint. *Transactions of the Association for Computational Linguistics*, 11:974–996.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Sohan Patnaik, Heril Changwal, Milan Aggarwal, Sumit Bhatia, Yaman Kumar, and Balaji Krishnamurthy. 2024. CABINET: Content relevance-based noise reduction for table question answering. In *The Twelfth International Conference on Learning Representations*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Abhilash Potluri, Fangyuan Xu, and Eunsol Choi. 2023. Concise answers to complex questions: Summarization of long-form answers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9709–9728, Toronto, Canada. Association for Computational Linguistics.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2018. Data-to-text generation with content selection and planning. In *AAAI Conference on Artificial Intelligence*.

Ratish Puduppully and Mirella Lapata. 2021. Data-to-text generation with macro planning. *Transactions of the Association for Computational Linguistics*, 9:510–527.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Yixuan Su, David Vandyke, Sihui Wang, Yimai Fang, and Nigel Collier. 2021. Plan-then-generate: Controlled data-to-text generation via planning. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 895–909, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiangru Tang, Yiming Zong, Jason Phang, Yilun Zhao, Wangchunshu Zhou, Arman Cohan, and Mark Gerstein. 2024. Struc-bench: Are large language models really good at generating complex structured data?

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

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 *Advances in Neural Information Processing Systems*.

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. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 602–631, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Fangyuan Xu, Yixiao Song, Mohit Iyyer, and Eunsol Choi. 2023a. A critical evaluation of evaluations for long-form question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3225–3245, Toronto, Canada. Association for Computational Linguistics.

Fangyuan Xu, Yixiao Song, Mohit Iyyer, and Eunsol Choi. 2023b. A critical evaluation of evaluations for long-form question answering. *arXiv preprint arXiv:2305.18201*.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 174–184, New York, NY, USA. Association for Computing Machinery.

Haoxuan You, Rui Sun, Zhecan Wang, Long Chen, Gengyu Wang, Hammad Ayyubi, Kai-Wei Chang, and Shih-Fu Chang. 2023. IdealGPT: Iteratively decomposing vision and language reasoning via large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11289–11303, Singapore. Association for Computational Linguistics.

Liangyu Zha, Junlin Zhou, Liyao Li, Rui Wang, Qingyi Huang, Saisai Yang, Jing Yuan, Changbao Su, Xiang Li, Aofeng Su, Tao Zhang, Chen Zhou, Kaizhe Shou, Miao Wang, Wufang Zhu, Guoshan Lu, Chao Ye, Yali Ye, Wentao Ye, Yiming Zhang, Xinglong Deng, Jie Xu, Haobo Wang, Gang Chen, and Junbo Zhao. 2023. Tablegpt: Towards unifying tables, nature language and commands into one gpt.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2024. Tablellama: Towards open large generalist models for tables.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. 2022a. MultiHiertt: Numerical reasoning over multi hierarchical tabular and textual data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6588–6600, Dublin, Ireland. Association for Computational Linguistics.

Yilun Zhao, Hongjun Liu, Yitao Long, Rui Zhang, Chen Zhao, and Arman Cohan. 2023a. Knowledgemath: Knowledge-intensive math word problem solving in finance domains.

Yilun Zhao, Yitao Long, Hongjun Liu, Linyong Nan, Lyuhao Chen, Ryo Kamoi, Yixin Liu, Xiangru Tang, Rui Zhang, and Arman Cohan. 2023b. Docmath-eval: Evaluating numerical reasoning capabilities of llms in understanding long documents with tabular data.

Yilun Zhao, Linyong Nan, Zhenting Qi, Rui Zhang, and Dragomir Radev. 2022b. ReasTAP: Injecting table reasoning skills during pre-training via synthetic reasoning examples. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9006–9018, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yilun Zhao, Zhenting Qi, Linyong Nan, Lorenzo Jaime Flores, and Dragomir Radev. 2023c. LoFT: Enhancing faithfulness and diversity for table-to-text generation via logic form control. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 554–561, Dubrovnik, Croatia. Association for Computational Linguistics.

Yilun Zhao, Zhenting Qi, Linyong Nan, Boyu Mi, Yixin Liu, Weijin Zou, Simeng Han, Ruizhe Chen, Xiangru Tang, Yumo Xu, Dragomir Radev, and Arman Cohan. 2023d. QTSumm: Query-focused summarization over tabular data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1157–1172, Singapore. Association for Computational Linguistics.

Yilun Zhao, Haowei Zhang, Shengyun Si, Linyong Nan, Xiangru Tang, and Arman Cohan. 2023e. Investigating table-to-text generation capabilities of large language models in real-world information seeking scenarios. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 160–175, Singapore. Association for Computational Linguistics.

Yilun Zhao, Chen Zhao, Linyong Nan, Zhenting Qi, Wenlin Zhang, Xiangru Tang, Boyu Mi, and Dragomir Radev. 2023f. RobuT: A systematic study of table QA robustness against human-annotated adversarial perturbations. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6064–6081, Toronto, Canada. Association for Computational Linguistics.

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

Alex Zhuang, Ge Zhang, Tianyu Zheng, Xinrun Du, Junjie Wang, Weiming Ren, Stephen W. Huang, Jie Fu, Xiang Yue, and Wenhu Chen. 2024. Structlm: Towards building generalist models for structured knowledge grounding.

A Appendix

Content Planning (Question Decomposition)

### Instruction:
Given a complex question about a table, your task is to assess the number of distinct facts the question seeks. If the question encompasses more than one fact, it necessitates deconstruction into a series of sequential sub-questions. This process involves breaking down the complexity to address each fact individually. On the other hand, if the question is focused on a single fact, your response should involve presenting the question as is, without further decomposition.

### Example Question:
Which song earned the highest points in the Bundesvision Song Contest 2010, and how many points did it secure?

### Example Response (in numbered list format):
1. Which song earned the highest points in the Bundesvision Song Contest 2010?
2. How many points did the song secure?

### Target Question:
{question}

### Target Response (in numbered list format):
1.

Figure 3: Given a complex question over the given table, the `content planner` decomposes the question into a series of step-by-step sub-questions.

System	**BL**	**R-L**	**ME**	**BS**	**T**$_{Acc}$	**ACU**
Blueprint	28.7	51.2	58.0	**90.2**	84.1	53.0
0-shot Direct	28.8	50.1	56.6	89.1	87.8	53.3
Chain-of-Thought	28.2	51.0	56.9	85.7	88.4	53.6
1-shot Direct	27.4	50.3	57.2	87.9	83.8	54.2
REFACTOR	26.2	53.6	57.2	87.4	**90.2**	54.8
Dater	29.8	**54.0**	59.4	88.2	86.3	**56.8**
TAPERA	29.5	53.4	58.2	86.1	87.2	55.2
wo. Table Reasoner	28.8	53.1	57.7	87.2	86.9	55.0
wo. Sub-answer Conv.	28.2	52.9	58.0	88.1	85.4	55.9
wo. Iterative Plan Refine.	28.7	53.4	56.2	85.9	86.3	55.3

Table 7: Automated evaluation results on the FETAQA test sets. **BL** denotes BLEU, **R-L** denotes ROUGE-L, **BS** denotes BERTScore, **ME** denotes METEOR, **T**$_{Acc}$ denotes TAPAS-Acc. We use **ACU** as the ranking indicator of model performance.

Executable Python Program Generation

### Instruction:
Given the table and a related question, your task is to develop an executable Python program to process the table data and provide the answer. The program should be designed to analyze the table contents and return the answer in string format if it is directly available within the table. If the table does not contain the necessary information to answer the question, the program should be programmed to return 'None'.

### Example:
Question: Which country has won the OGAE Video Contest multiple times?
Table Title: OGAE Video Contest - Winners
Table:

```
   "header": ["Year", "Country", "Video", "Performer", "Points", "City"],
   "rows": [
       ["2003", "France", "\"Fan\"", "Pascal Obispo", "122", "Turkey Istanbul"],
       ["2004", "Portugal", "\"Cavaleiro Monge\"", "Mariza", "133", "France Fontainebleau"],
      ["2005", "Ukraine", "\"I Will Forget You\"", "Svetlana Loboda", "171", "Portugal Lisbon"],
       ["2006", "Italy", "\"Contromano\"", "Nek", "106", "Turkey Izmir"],
       ["2007", "Russia", "\"LML\"", "Via Gra", "198", "Italy Florence"],
       ["2008", "Russia", "\"Potselui\"", "Via Gra", "140", "Russia Moscow"]
   ]
```

### Example Response:

~~~
```python
def multiple_winners_in_OGAE_contest(table):
   from collections import Counter

   # Extract the column for countries
   country_column_index = table["header"].index("City")
   countries = [row[country_column_index] for row in table["rows"]]

   # Count the victories for each country
   victory_count = Counter(countries)

   # Filter and return countries with multiple victories
   multiple_victories = [country for country, count in victory_count.items() if count > 1]
   return ', '.join(multiple_victories) if multiple_victories else None
```
~~~

### Target:
Question: {question}
Table Title: {table_title}
Table: {table}

### Target Response:

~~~
```python
~~~

Figure 4: Given a sub-question in QA-based plan and a table, the `execution-based table reasoner` generates an executable Python program as an intermediate reasoning step to enhance table reasoning.

Sub-Answer Generation

### Instruction:
Given a specific question along with its corresponding answer, your objective is to construct a comprehensive, well-structured response in a single, elongated sentence. The response should be formulated directly based on the provided answer, ensuring that it thoroughly addresses the query.

### Example QA:
Question: Which country has won the OGAE Video Contest multiple times?
Answer: Russia, France, and Belgium.

### Example Response:
Russia, France, and Belgium have each won the OGAE Video Contest multiple times.

### Target QA:
Question: {question}
Answer: {answer}

### Target Response:

Figure 5: Given the question and execution result of the generated Python program, the `answer generator` generate a sentence-long answer.

Final Answer Generation

### Instruction:
Given a question and a list of relevant facts, each containing part of the information needed to answer the question, your task is to synthesize these facts into a coherent, long sentence that fully addresses the question. The goal is to integrate the individual pieces of information from the list seamlessly, creating a comprehensive response that encapsulates all aspects of the question in a clear and concise manner.

### Example Question:
Which country has won the OGAE Video Contest multiple times, and what were the corresponding years, winning songs and points scored?

### Example QA-based Plan:
Sub-Question 1: Which country has won the OGAE Video Contest multiple times?
Sub-Answer 1: Russia, France, and Belgium have each won the OGAE Video Contest multiple times.

Sub-Question 2: What were the corresponding years of their victories?
Sub-Answer 2: Russia won the contest in 2007, 2008, and 2009, France in 2003, 2011, and 2014, and Belgium in 2013 and 2017.

Sub-Question 3: What were the winning songs for each country?
Sub-Answer 3: Russia's winning songs were "LML" in 2007, "Potselui" in 2008, and "Karma" in 2009; France's were "Fan" in 2003, "Lonely Lisa" in 2011, and "Tourner dans le vide" in 2014; Belgium's were "Papaoutai" in 2013 and "Mud Blood" in 2017.

### Example Response:
Russia, France, and Belgium have each won the OGAE Video Contest multiple times. Specifically, Russia triumphed in 2007 with "LML" scoring 198 points, in 2008 with "Potselui" scoring 140 points, and in 2009 with "Karma" scoring 142 points. France secured victories in 2003 with "Fan" scoring 122 points, in 2011 with "Lonely Lisa" scoring 96 points, and in 2014 with "Tourner dans le vide" scoring 141 points. Belgium won in 2013 with "Papaoutai" scoring 144 points and again in 2017 with "Mud Blood" scoring 184 points.

### Target Question:
{question}

### Target QA-based Plan:
{qa_plan}

### Target Response:

Figure 6: The `answer generator` generates the final answer based on the finalized QA-based plan.

Failure Scenarios	**Example**	**Explanation**
System reveals additional reasoning-intensive information	**Question:** Compare the profit information of Sinopec Group and Apple in 2012. **Ground-truth answer:** Sinopec Group earned $6,205 million in profit in 2012, while Apple earned $57,501 million in profit, which is much more than Sinopec Group.	TaPERA excels at integrating additional, reasoning-intensive information not present in the ground-truth answers, as demonstrated by the inclusion of "approximately 9 times greater than that of Sinopec Group" in the example. This capability is attributed to its effective content planning through a QA-based plan and enhanced reasoning mechanisms facilitated by the Table Reasoner. Nonetheless, the capacity of current evaluation metrics to capture such supplementary information is limited.
Ground truth answer is redundant	**Question:** How many aircraft models were first introduced between 1980 and 1985, and what are their build years? **Ground Truth:** The number of aircraft models that were initially launched within the timeframe extending from the year 1980 up to and including the year 1985 amounts to a total of three distinct models. Specifically, the construction years for these models are identified as the years 1978, 1979, and 1971 respectively.	We found that the answers generated by TaPERA are more concise compared to the ground truth and other LLM-based models, yet they retain all essential information relevant to the question. This finding is supported by Table 2 in the manuscript. The "consistency" feature of TaPERA is key to this achievement, allowing it to produce answers that not only align with the QA-based plan but also avoid the inclusion of any irrelevant or redundant details. In real-world applications, this quality is particularly valued, as human users often prefer answers that are both concise and accurate.
Question is ambiguous	**Question:** How did countries that incorporate dual languages in their songs perform in this competition?	**Explanation:** The question is ambiguous and can lead to multiple valid answers. In such cases, LLM-based systems tend to generate more words than TaPERA, increasing the likelihood of overlapping information between the model output and the ground truth. However, recent studies have shown that lengthy answers for long-form QA are not always necessary, and humans generally prefer concise answers that still fully address the question.
Ground truth answer is misaligned with the question		**Explanation:** The annotated ground truth fails to fully or accurately address the question.

Table 8: Failure cases analysis of automated evaluation systems for the LFTQA task.