H-STAR: LLM-driven Hybrid SQL-Text Adaptive Reasoning on Tables

Nikhil Abhyankar[1]**, Vivek Gupta**[2]**, Dan Roth**[3]**, Chandan K. Reddy**[1]
[1]Virginia Tech, [2]Arizona State University, [3]University of Pennsylvania
nikhilsa@vt.edu, vgupt140@asu.edu, danroth@seas.upenn.edu, reddy@cs.vt.edu

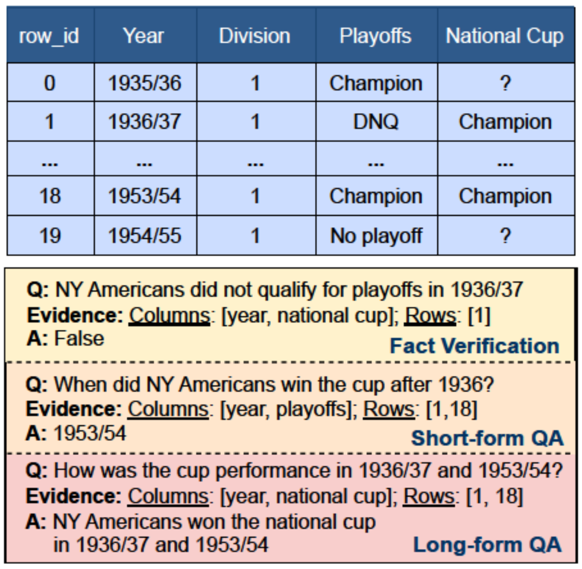
Abstract

Tabular reasoning involves interpreting natural language queries about tabular data, which presents a unique challenge of combining language understanding with structured data analysis. Existing methods employ either textual reasoning, which excels in semantic interpretation but struggles with mathematical operations, or symbolic reasoning, which handles computations well but lacks semantic understanding. This paper introduces a novel algorithm H-STAR that integrates both symbolic and semantic (textual) approaches in a two-stage process to address these limitations. H-STAR employs: (1) step-wise table extraction using 'multi-view' column retrieval followed by row extraction, and (2) adaptive reasoning that adapts reasoning strategies based on question types, utilizing semantic reasoning for direct lookup and complex lexical queries while augmenting textual reasoning with symbolic reasoning support for quantitative and logical tasks. Our extensive experiments demonstrate that H-STAR significantly outperforms state-of-the-art methods across three tabular question-answering (QA) and fact-verification datasets, underscoring its effectiveness and efficiency.[1]

1 Introduction

Tabular data are among the most widely used formats for storing structured information in real-world applications. Tabular reasoning presents an inherently challenging problem, requiring logical, mathematical, and textual reasoning over unstructured queries and structured tables (Ye et al., 2023). Thus, understanding and inferring tabular data has become a significant area of research in natural language processing. Tabular reasoning tasks (Figure 1), such as table-based question answering (Pasupat and Liang, 2015; Nan et al., 2022) and table-based fact verification (Chen et al., 2019), have been extensively explored in the past.

[1]The code and data are available at: https://github.com/nikhilsab/H-STAR



row_id	Year	Division	Playoffs	National Cup
0	1935/36	1	Champion	?
1	1936/37	1	DNQ	Champion
...	...	...	...	...
18	1953/54	1	Champion	Champion
19	1954/55	1	No playoff	?

Q: NY Americans did not qualify for playoffs in 1936/37
Evidence: Columns: [year, national cup]; Rows: [1]
A: False
Fact Verification

Q: When did NY Americans win the cup after 1936?
Evidence: Columns: [year, playoffs]; Rows: [1,18]
A: 1953/54
Short-form QA

Q: How was the cup performance in 1936/37 and 1953/54?
Evidence: Columns: [year, national cup]; Rows: [1, 18]
A: NY Americans won the national cup in 1936/37 and 1953/54
Long-form QA

Figure 1: An illustration of different tabular reasoning tasks (a) **Fact-verification**, (b) **Short-form QA**, and (c) **Long-form QA**. For each task, question Q is paired with its answer A, which varies by task. Evidence shows the relevant columns and rows needed to answer the question.

Advancements in Large Language Models (LLMs) have led to better performance across various tasks through carefully crafted prompts. In the domain of table reasoning, symbolic approaches such as Program of Thought (Chen et al., 2023) and textual methods such as Chain of Thought (CoT) (Wei et al., 2022; Brown et al., 2020) have been explored. However, when applied individually (see Figure 2) these methods often struggle due to the complexities imposed by the intricate mix of numerical, temporal, and textual data, coupled with complex table structures (Shi et al., 2023; Liu et al., 2023; Sui et al., 2024). Textual reasoning excels in natural language understanding but often misinterprets table structures and struggles with quantitative reasoning. In contrast, SQL-based approaches are strong in quantitative problem solving, but perform poorly on noisy or unstructured inputs (Liu et al., 2023). As a result, recent techniques (Cheng

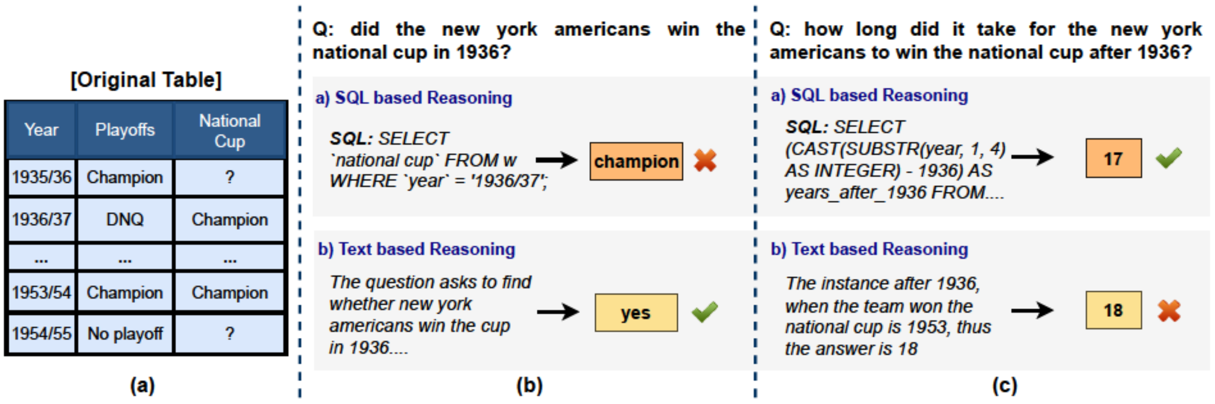

[Original Table]
Year
Playoffs
National Cup
1935/36
Champion
?
1936/37
DNQ
Champion
...
1953/54
Champion
Champion
1954/55
No playoff
?
(a)
Q: did the new york americans win the national cup in 1936?
a) SQL based Reasoning
SQL: SELECT `national cup` FROM w WHERE `year` = '1936/37';
champion
b) Text based Reasoning
The question asks to find whether new york americans win the cup in 1936....
yes
(b)
Q: how long did it take for the new york americans to win the national cup after 1936?
a) SQL based Reasoning
SQL: SELECT (CAST(SUBSTR(year, 1, 4) AS INTEGER) - 1936) AS years_after_1936 FROM....
17
b) Text based Reasoning
The instance after 1936, when the team won the national cup is 1953, thus the answer is 18
18
(c)

Figure 2: **An illustration highlighting the complexity of table reasoning and the need for an integrated approach**: (a) Original table, (b) Symbolic reasoning misinterprets the question and returns a value instead of a yes/no response, and (c) Text-based approach fails to solve a math question correctly, leading to an incorrect answer.

et al., 2022; Ye et al., 2023; Wang et al., 2023; Nahid and Rafiei, 2024) that rely solely on textual or symbolic reasoning suffer from similar limitations. Therefore, it is crucial to explore the question: *How can **symbolic and textual approaches be integrated** into a hybrid method for tabular reasoning that **leverages their complementary strengths while mitigating their individual limitations**?*

In this paper, we introduce H-STAR which combines symbolic and textual reasoning abilities of LLMs, achieving the *best of both worlds*. H-STAR decomposes the table reasoning task into two subtasks: **(a) Table Extraction** and **(b) Adaptive Reasoning**. In the table extraction phase, we employ a step-by-step 'multi-view' chain, first identifying relevant columns using the original table and its transposed form. The filtered table is then used for row extraction, contrary to other approaches, which use the original table for both row and column selection (Ye et al., 2023; Nahid and Rafiei, 2024). Focusing on relevant table cells helps LLMs by providing the right context, thus reducing hallucinations in reasoning. We employ a combination of SQL-based and text-based techniques in H-STAR's table extraction phase. In the adaptive reasoning phase, we use the language comprehension capabilities of LLMs to guide them with few-shot examples to decide when to support semantic methods with symbolic (SQL) approaches. H-STAR employs semantic reasoning universally, using it exclusively for direct lookup, common sense, and complex lexical queries while using an additional SQL step for quantitative, mathematical, and logical tasks. This strategy optimizes performance on diverse question types, supporting text comprehension with computation when required.

We demonstrate the efficacy of H-STAR across three tabular benchmarks involving table QA and table fact verification. Our experiments show that using the combined approach of H-STAR, aligned with multi-view extraction and adaptive reasoning outperforms all prior state-of-the-art approaches. Our analysis demonstrates that H-STAR is robust and generalizable across LLMs, showcasing its superior reasoning capabilities across various tasks. Our main contributions are as follows:

(a) We introduce H-STAR, a two-step method unifying table extraction with adaptive reasoning to enhance performance on tabular reasoning. It is a hybrid approach that effectively integrates symbolic (SQL logic) and semantic (textual) methods.

(b) Through extensive experiments, we show that H-STAR's hybrid approach surpasses state-of-the-art methods that rely solely on either symbolic or semantic techniques, especially when reasoning over larger tables.

(c) Our ablation study and analysis highlight H-STAR's effectiveness in table extraction and reasoning across various models and datasets, underscoring the significance of each stage in enhancing overall performance.

The code and additional resources are available at: `https://hstar-llm.github.io/`.

2 H-STAR Approach

H-STAR decomposes the table reasoning task into extraction and reasoning stages, combining LLM's textual comprehension with symbolic reasoning. It converts the original table ($\mathtt{T}$) into a query-specific table ($\mathtt{T}_{\mathtt{CR}}$) (refer Algorithm 1). Figure 3 illustrates our H-STAR approach. Unlike DATER (Ye et al., 2023) and TabQSLify (Nahid and Rafiei, 2024), which use textual reasoning and text-to-SQL respectively, H-STAR integrates both reasoning types in a complementary manner for table extraction. Our reasoning step dynamically uses symbolic techniques alongside semantic methods for quantitative questions, overcoming the limits of pure textual reasoning. H-STAR operates in two main stages: *1) Table Extraction* and *2) Adaptive Reasoning*. Implementation details, including prompts, input table formats, and hyperparameters, are provided in Appendix C and D.

2.1 Table Extraction

Two-dimensional tables consist of columns and rows. Table extraction involves a two-step reasoning chain: **(1)** column extraction and **(2)** row extraction (see Figure 3). As illustrated in Algorithm 1, for a given table T, table extraction returns a table T_{CR} (lines 1-8) based on the input query Q.

Column Extraction. As shown in Algorithm 1, H-STAR uses a 'multi-view' technique, first using the original table T (line 1) followed by its transposed form T^T (line 2) for column extraction. Changing the 'view' acts as a verifier accounting for any information missed due to the table structure, thus improving the overall performance (see Appendix A.3). In Figure 3, given an input table T and the question Q (*'How long did it take the New York Americans to win the National Cup after 1936?'*), the LLM employs $\texttt{col}_{\texttt{sql}}$ to generate an SQL query to extract columns *'year'* (C_1). This is followed by the text-based step $\texttt{col}_{\texttt{text}}$ which processes table T^T to return columns C_2 (*'year', 'national cup'*). This step captures the 'national cup' information relevant to answering the query, which is missed by $\texttt{col}_{\texttt{sql}}$. Table T is then filtered for columns *'year', 'national cup'* (C') derived from C_1 and C_2 to obtain table T_c (lines 3,4).

Row Extraction. After obtaining the filtered table T_C with relevant columns (Algorithm 1, line 4), H-STAR proceeds to the row extraction phase culminating in the query-specific table T_{CR} (lines 5-8). The filtered table T_C has fewer tokens than T, fitting within the token limit and enabling more efficient row extraction. As shown in Figure 3, the SQL query generated by $\texttt{row}_{\texttt{sql}}$ on the column-filtered table T_C extracts the relevant rows R_1 (*'row 18'*). Although it returns the appropriate row, it is not sufficient to answer the question. To overcome these limitations, we use text-based row verification $\texttt{row}_{\texttt{text}}$ to obtain the missing rows, ultimately retrieving *'row 1'* and *'row 18'* (R_2). Combining R_1 and R_2 gives the relevant rows R' (*'row 1', 'row 18'*), resulting in the final table T_{CR}.

Algorithm 1 H-STAR

```
Input: (T, Q) → Table - Question pair
Output: $\hat{A}$ → Predicted Answer
    Stage A: Table Extraction
    Column Extraction
 1:    $C_1 \leftarrow \texttt{col}_{\texttt{sql}}(T, Q)$          ▷ SQL approach
 2:    $C_2 \leftarrow \texttt{col}_{\texttt{text}}(T^T, Q)$       ▷ Text approach
    ▷ $T^T$ refers to the transposed table
 3:    $C' \leftarrow C_1 \cup C_2$
 4:    $T_C \leftarrow T.\texttt{filter}(C')$
    ▷ Filter table T with columns $C'$ to get $T_C$
    Row Extraction
 5:    $R_1 \leftarrow \texttt{row}_{\texttt{sql}}(T_C, Q)$        ▷ SQL approach
 6:    $R_2 \leftarrow \texttt{row}_{\texttt{text}}(T_C, Q, R_1)$   ▷ Text approach
 7:    $R' \leftarrow R_1 \cup R_2$
 8:    $T_{CR} \leftarrow T_C.\texttt{filter}(R')$
    ▷ Filter table $T_C$ with rows $R'$ to get $T_{CR}$
    Stage B: Adaptive Reasoning
 9: if $\texttt{math}(Q) == \texttt{True}$ then
10:     $E_v \leftarrow \texttt{f}_{\texttt{sql}}(T_{CR}, Q)$
11:     $\hat{A} \leftarrow \texttt{f}_{\texttt{text}}(T_{CR}, Q, E_v)$
        ▷ Use SQL generated $E_v$ with $T_{CR}$ to generate $\hat{A}$.
12: else
13:     $\hat{A} \leftarrow \texttt{f}_{\texttt{text}}(T_{CR}, Q)$
        ▷ Use only the table $T_{CR}$ to generate $\hat{A}$
14: end if
15: return $\hat{A}$
```

2.2 Adaptive Reasoning

We propose an adaptive reasoning framework that harnesses the strengths of textual reasoning while mitigating its quantitative limitations through symbolic reasoning. First, we prompt LLMs to leverage their language understanding capabilities to evaluate the query requirements. We apply symbolic reasoning for queries necessitating quantitative analysis and use the generated output (E_v) along with the table (T_{CR}) as the input to the final textual reasoning step. For instance, in Figure 3, when faced with a question like *'How long did it take the New York Americans to win the National Cup after 1936?'*, LLM needs to calculate the answer, and our pipeline prioritizes symbolic reasoning ($\texttt{f}_{\texttt{sql}}$) through SQL-generated code. This code is executed on a SQL engine, enabling precise answers from the table T_{CR}. The output from this SQL-based evaluation E_v serves as additional evidence, supporting the final reasoning step ($\texttt{f}_{\texttt{text}}$). By explicitly integrating this quantitative approach, *our algorithm effectively addresses LLMs' limitations in handling quantitative reasoning*, thereby

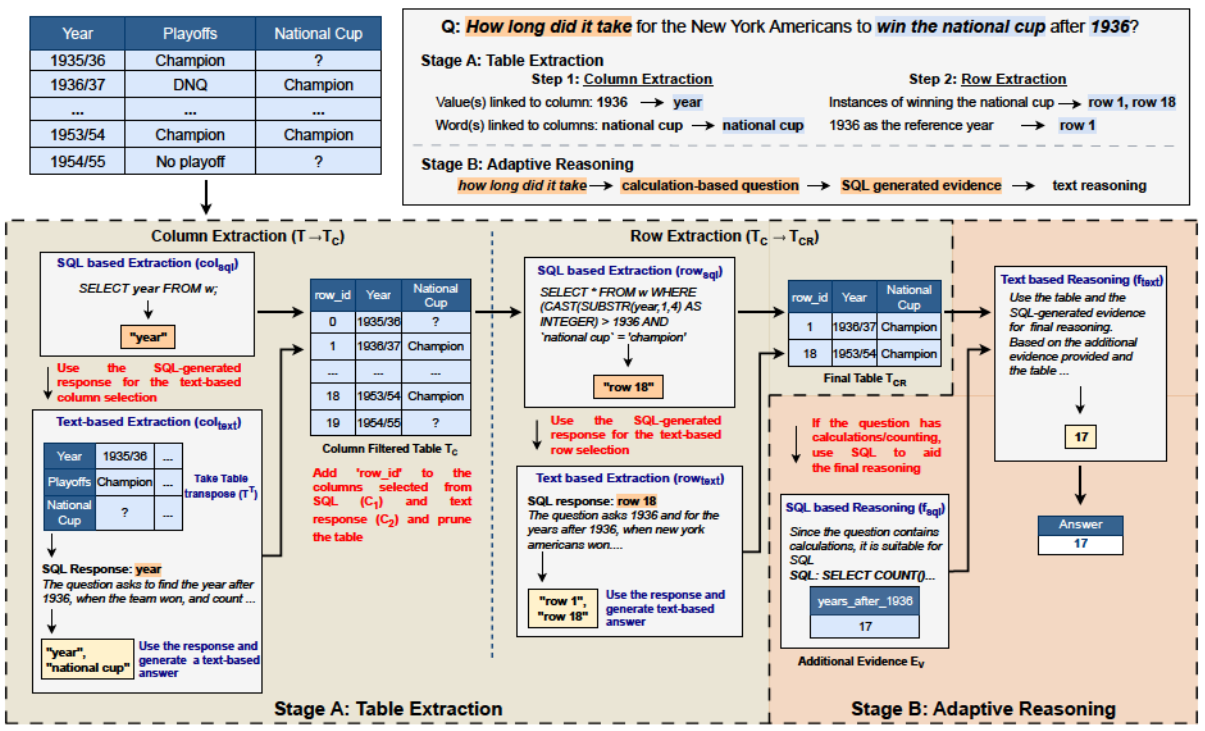

Year | Playoffs | National Cup
1935/36 | Champion | ?
1936/37 | DNQ | Champion
1953/54 | Champion | Champion
1954/55 | No playoff | ?
Q: How long did it take for the New York Americans to win the national cup after 1936?
Stage A: Table Extraction
Step 1: Column Extraction
Value(s) linked to column: 1936 → year
Word(s) linked to columns: national cup → national cup
Step 2: Row Extraction
Instances of winning the national cup → row 1, row 18
1936 as the reference year → row 1
Stage B: Adaptive Reasoning
how long did it take → calculation-based question → SQL generated evidence → text reasoning
Column Extraction (T → T_C)
SQL based Extraction (col_sql)
SELECT year FROM w;
"year"
Use the SQL-generated response for the text-based column selection
Text-based Extraction (col_text)
Take Table transpose (T^T)
SQL Response: year
The question asks to find the year after 1936, when the team won, and count ...
"year", "national cup"
Use the response and generate a text-based answer
row_id | Year | National Cup
0 | 1935/36 | ?
1 | 1936/37 | Champion
18 | 1953/54 | Champion
19 | 1954/55 | ?
Column Filtered Table T_C
Add 'row_id' to the columns selected from SQL (C_1) and text response (C_2) and prune the table
Row Extraction (T_C → T_CR)
SQL based Extraction (row_sql)
SELECT * FROM w WHERE (CAST(SUBSTR(year,1,4) AS INTEGER) > 1936 AND `national cup` = 'champion'
"row 18"
Use the SQL-generated response for the text-based row selection
Text based Extraction (row_text)
SQL response: row 18
The question asks 1936 and for the years after 1936, when new york americans won....
"row 1", "row 18"
Use the response and generate text-based answer
Final Table T_CR
If the question has calculations/counting, use SQL to aid the final reasoning
SQL based Reasoning (f_sql)
Since the question contains calculations, it is suitable for SQL
SQL: SELECT COUNT()...
years_after_1936
17
Additional Evidence E_V
Text based Reasoning (f_text)
Use the table and the SQL-generated evidence for final reasoning. Based on the additional evidence provided and the table ...
17
Answer
17
Stage A: Table Extraction
Stage B: Adaptive Reasoning

Figure 3: **An overview of H-STAR**, consisting of a combination of code generation and text-based verification. Given a complex table and its question, H-STAR answers using (a) **Table Extraction:** extracts the question-specific table from the original by first selecting the columns followed by rows. (b) **Adaptive Reasoning**: when the question has any mathematical component, it generates an additional table using SQL used in the textual reasoning step.

improving the accuracy and robustness when tackling questions that demand numerical reasoning.

3 Experiments

Benchmark Datasets. We evaluate our method on three datasets covering fact verification, short-form, and long-form question-answering tasks using in-context examples. (a) TabFact (Chen et al., 2019): A fact verification benchmark utilizing Wikipedia tables. We evaluated the test-small set, containing 2,024 statements and 298 tables, with each statement labeled Entailed ("True") or Refuted ("False"). (b) WikiTQ (Pasupat and Liang, 2015): The WikiTableQuestions (WikiTQ) dataset involves question-answering tasks over semi-structured Wikipedia tables. It includes a standard test set with 4,344 table-question pairs. (c) FeTaQA (Nan et al., 2022): FeTaQA (Free-Form Table Question Answering) comprises free-form questions requiring synthesis from multiple table sections for responses. It demands advanced reasoning and profound tabular data understanding, evaluated on a test set of 2003 samples.

Evaluation Metrics. We tailor our evaluation metrics based on the task and dataset characteristics. For fact verification data sets such as TabFact, we evaluated performance using *binary classification accuracy*. For short-form question-answering datasets such as WikiTQ, we assess accuracy by measuring the *exact match* between predicted outputs and the gold-standard answers. For more complex tasks in FeTaQA, which involve long-form question answering, we evaluate performance using *ROUGE-1*, *ROUGE-2*, and *ROUGE-L* metrics (Lin, 2004), comparing predicted outputs with long-form answers.

LLM Models. In our research, we use state-of-the-art large language models (LLM) such as Gemini-1.5-Flash (Reid et al., 2024), PaLM-2 (Anil et al., 2023), GPT-3.5-Turbo, GPT-4o-mini (OpenAI, 2023), and the open-source Llama-3-70B (Dubey et al., 2024) for table reasoning tasks. Our model inputs include in-context examples, the table, and the question for each step of the pipeline. All baselines with PALM-2 are from the propriety PALM-2 model, which is different from the public PALM-2 API (used in H-STAR and TabSQLify).

Baseline Methods. We compare our method with (a) generic reasoning based on language models, and (b) table-manipulation reasoning based on language models.

(a) Generic Reasoning. These methods direct the LLM to carry out the required downstream task based on the information from the table and the input question. They include End-to-End QA, which offers only the table and the question, and Few-Shot QA, which involves a few examples with the table-question-answer triplet alongside the table and the question. Chain-of-Thought prompts the LLMs to provide a supporting reasoning chain that leads to the answer. **(b) Table Manipulation.** These techniques involve several steps, with the initial stage dedicated to automatically pre-processing the table for the reasoning task. BINDER utilizes in-context examples to produce SQL or Python code containing LLM API calls to generate answers. DATER directs the language model to break down tables and questions, applying reasoning to the decomposed tables based on sub-questions. Chain-of-Table employs the table as an intermediate output in its reasoning process, iterating through tasks until the final answer is obtained. TabSQLify uses SQL to trim the table and then reasons based on the reduced table.

3.1 Main Results

Table 1 compares the performance of different methods on the TabFact, and WikiTQ datasets, across GPT-3.5-Turbo and PaLM-2. This comparison involves evaluating against generic reasoning, table manipulation techniques, and H-STAR. Appendix A.1 offers a comprehensive analysis of the FeTaQA dataset, including a comparison with baselines, human evaluation, and qualitative analysis.

	GPT-3.5-Turbo		PaLM-2	
	TabFact	**WikiTQ**	**TabFact**	**WikiTQ**
Generic Reasoning				
End-to-End QA	70.45	51.84	77.92	60.59
Few-shot QA	71.54	52.56	78.06	60.33
CoT	65.37	53.48	79.05	60.43
Table Manipulation				
BINDER	79.17	56.74	76.98	54.88
DATER	78.01	52.90	84.63	61.48
Chain-of-Table*	80.20	59.94	**86.61**	67.31
TabSQLify	79.50	64.70	79.78	55.78
H-STAR	**85.03**	**69.56**	86.51	**68.62**

Table 1: **Comparison of various methods across datasets.** The results are reported from the Chain-of-Table paper for fair comparison. * Chain-of-Table uses a proprietary PaLM-2 model that is better than the publicly available version.

Analysis. (a) On WikiTQ dataset, H-STAR surpasses all baselines with both GPT-3.5-Turbo and PaLM-2 models. H-STAR achieves an accuracy of 69.56% on WikiTQ with GPT-3.5-Turbo, marking an improvement of 17.72% over the vanilla GPT-3.5-Turbo model. (b) On the TabFact dataset, H-STAR outperforms all methods with GPT-3.5-Turbo, attaining an accuracy of 85.03%, which is a 14.58% improvement over the vanilla model. With PaLM-2, H-STAR achieves comparable performance with Chain-of-Table on TabFact while improving over it by 1.3% on WikiTQ.

Comparison Across Methods. Table 2 compares H-STAR with ReAcTable (Zhang et al., 2023) and SYNTQA (Zhang et al., 2024b), which use both textual and symbolic approaches for table reasoning. SYNTQA (GPT) uses the GPT model to decide between semantic and symbolic approaches, applying the chosen method for reasoning based on its selection. Additionally, it is compared with ALTER (Zhang et al., 2024a), which augments both queries and table data to facilitate reasoning and E5 (Zhang et al., 2024c), leveraging code generation capabilities of LLMs for table extraction followed by reasoning.

	TabFact	**WikiTQ**
ReAcTable	73.1	52.5
E5	75.6	50.9
ALTER	84.3	67.4
SYNTQA (GPT)	-	65.2
H-STAR	**85.0**	**69.6**

Table 2: Comparison of various methods across datasets on GPT-3.5-Turbo.

Analysis. H-STAR outperforms methods that use symbolic and textual approaches, such as ReAcTable, SYNTQA(GPT), E5, and ALTER, showing the effectiveness of its complementary reasoning in a two-stage process. Appendix A.2 presents a comprehensive comparison of methods, including pre-trained models, fine-tuned architectures, and state-of-the-art LLM-based approaches. Our evaluation demonstrates that H-STAR consistently achieves superior performance across all benchmark metrics compared to all the baselines.

Performance Across LLMs. Table 3 compares the performance of more advanced models like Gemini-1.5-Flash, GPT-4o-mini, and the open-source Llama-3-70B on TabFact and WikiTQ datasets. The results emphasize H-STAR's generalizability across different models and consistent improvements across datasets. Even advanced models show significant benefits by using H-STAR.

	GPT-4o-mini		Gemini-1.5		Llama-3	
	TF	**WTQ**	**TF**	**WTQ**	**TF**	**WTQ**
Generic Reasoning						
End-to-End QA	73.22	59.43	81.12	58.47	78.41	57.89
CoT	75.99	64.31	79.99	64.11	75.34	65.49
Table Manipulation						
TabSQLify	78.30	68.74	79.50	63.92	60.70	66.85
Chain-of-Table	85.09	68.53	86.95	70.05	85.86	70.76
H-STAR	**89.42**	**74.93**	**89.08**	**73.14**	**89.23**	**75.76**

Table 3: Comparison of various models across datasets. **TF**: TabFact; **WTQ**: WikiTQ

Analysis. (a) H-STAR shows much higher accuracy for GPT-4o-mini, scoring 89.42% on TabFact and 74.93% on WikiTQ, compared to the End-to-End QA and Chain of Thought methods, which achieve 73.22% and 75.99% on TabFact and 59.43% and 64.31% on WikiTQ. This marks a 16.2% improvement on TabFact and 15.5% on WikiTQ. (b) For Gemini-1.5-Flash, accuracy rises from 81.12% and 58.47% respectively to 89.08% on TabFact and 73.14% on WikiTQ. (c) Similarly, for Llama-3-70B, H-STAR improves the performance across both the datasets by a margin of 11% on TabFact and 18% on WikiTQ, showing consistent gains across models and datasets.

3.2 Efficiency Analysis

Efficiency of Table Extraction. Stage-1 in H-STAR involves the extraction of the table most relevant to the question. Our table extraction method uses a two-step chain to select the relevant columns followed by the rows. Figure 4 compares the number of average table cells for the extracted table for H-STAR, H-STAR without row extraction, and H-STAR without column extraction with other baselines.

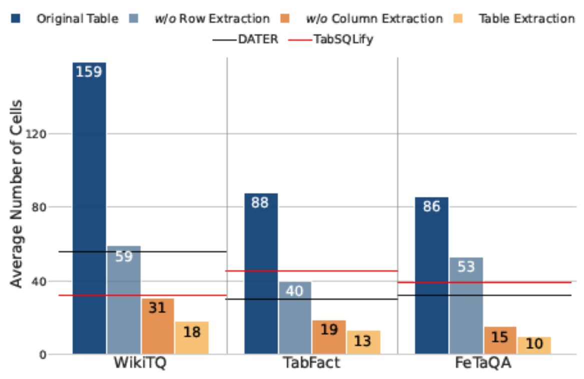


Figure 4: Comparison of average table cells in the final table.

Analysis. H-STAR employs a structured, step-by-step process: it first extracts query-relevant columns before selecting relevant rows. This progressive filtering minimizes token usage at each stage, thereby reducing computational overhead. As a result, H-STAR significantly reduces the average number of processed table cells—dropping from 159 to 18 on the WikiTQ dataset, from 88 to 13 on TabFact, and from 86 to 10 on FeTaQA. In contrast, alternative table manipulation methods, such as TabSQLify and DATER, yield considerably higher values. By effectively filtering out irrelevant information, H-STAR enables LLMs to focus on the *right evidence for right reasoning*, demonstrating superior efficiency across all datasets.

Efficiency on Longer Tables. Table-CoT (Chen, 2023) highlights that longer table size, with tables exceeding 30 rows, is a significant cause of erroneous generations. This observation is supported by a decline in LLM performance as the number of tokens in the table increases. In Table 4, we classify LLM performance based on total tokens into three groups - small (< 2000 tokens), medium (2000 to 4000 tokens), and large (> 4000 tokens) and compare H-STAR with table manipulation like methods BINDER, DATER, Chain-of-Table, and TabSQLify.

Method	**Small**	**Medium**	**Large**
BINDER	56.54	25.13	6.41
DATER	62.50	42.34	34.62
Chain-of-Table	68.13	52.25	44.87
TabSQLify	68.15	57.91	52.34
H-STAR	**71.64**	**65.20**	**64.84**

Table 4: Performance of various methods on different table sizes on WikiTQ.

Analysis. The results presented in Table 4 highlight the challenges LLMs face when reasoning over longer tables. Our findings confirm previous research, showing a significant decline in LLM performance as table sizes increase. While other table manipulation methods suffer from this scaling issue, H-STAR maintains consistent performance across all table sizes. This success is attributed to H-STAR 's efficient table extraction process, which reduces the table size by removing irrelevant data that would otherwise act as noise and hinder reasoning.

Resource Efficiency. In Table 5, we analyze H-STAR by examining the total number of samples generated by LLMs. BINDER and DATER employ self-consistency techniques to refine their results, while Chain-of-Table follows an iterative sampling process. Specifically, BINDER generates 50 Neural-SQL samples using self-consistency, whereas DATER applies self-consistency at each step, producing 100 samples. In contrast, Chain-of-Table adopts a more resource-efficient approach, generating 25 samples across three steps: 'Dynamic Plan', 'Generate Args', and 'Query'. TabSQLify generates the fewest samples, with a single generation for both table decomposition and query steps. However, relying on single outputs can reduce accuracy when the model fails to return a valid answer. To mitigate this, H-STAR integrates self-consistency by generating two outputs per stage, ensuring output validity. Each step, except 'Query', involves two generations for both SQL and text, totaling four steps for column and row retrieval. These additional steps act as safeguards against unreliable LLM outputs. Overall, H-STAR maintains efficiency, requiring only 6–10 sample generations.

Method	# samples / step	Total # samples
BINDER	Neural SQL: 50	50
DATER	Decompose Table: 40 Generate Cloze: 20 Generate SQL: 20 Query: 20	100
Chain-of-Table	Dynamic Plan ≤ 5 Generate Args ≤ 19 Query: 1	≤ 25
TabSQLify	Table Decompose: 1 Query: 1	2
H-STAR	Column Extraction: 2-4 Row Extraction: 2-4 Query: 2	6-10

Table 5: Number of generated samples for different methods.

3.3 Error Analysis

H-STAR. The disjoint, step-wise nature of H-STAR enables identifying and analyzing failures. Figure 5 shows 100 randomly selected instances from the Tabfact and WikiTQ datasets where H-STAR returns incorrect answers. In this study, 'Missing Columns' and 'Missing Rows' refer to missing necessary columns and rows, respectively. 'Incorrect Reasoning' occurs when H-STAR extracts the correct table, but LLM fails to produce the correct answer. 'Incorrect Annotations' include semantically identical answers in different formats, ambiguous questions, and incorrect gold answers. Figure 5 shows that for 100 TabFact failures, 2% are missing columns, 9% missing rows, 79% incorrect LLM reasoning, and 10% incorrect annotations. For WikiTQ, 6% are missing columns, 17% missing rows, 54% incorrect reasoning, and 23% incorrect annotations.

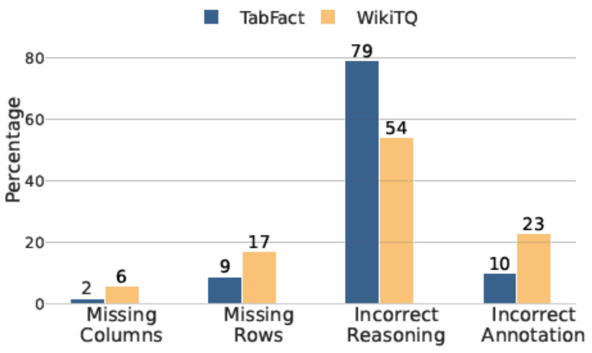


Figure 5: Error distribution on 100 error samples across datasets for H-STAR (GPT-3.5-Turbo).

Analysis. Fewer errors in column and row extraction indicate that H-STAR effectively retrieves the necessary table data. This improvement allows us to "shift left" by addressing issues earlier in the pipeline, enhancing overall performance. The higher percentage of errors in reasoning does not reflect poorly on the LLM; instead, it underscores the effectiveness of our table extraction process.

H-STAR vs Others. Figure 6 compares errors in H-STAR, with TabSQLify, and BINDER on 100 WikiTQ samples where TabSQLify fails. TabSQLify extracts wrong tables for 62 out of 100 (6 missing columns, 56 missing rows). Out of the 38 samples that remain, it incorrectly reasons on 29 of the samples (76%). BINDER does not handle table extraction and instead relies on generating multiple neural SQL queries for final reasoning. However, it fails to reason correctly in 70 out of 100 instances.

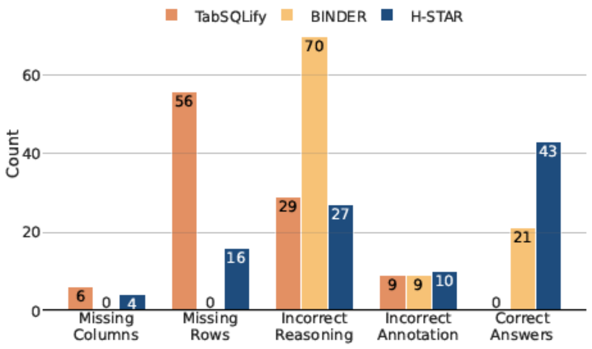


Figure 6: Analysis of error types in 100 samples from WikiTQ where TabSQLify fails. **Note**: As we move from left to right, the total samples decrease for each stage in the pipeline.

Analysis. In contrast, H-STAR proves more effective in both table extraction and reasoning, generating correct answers on 43 samples. Additionally, it demonstrates fewer errors in table extraction (4 missed columns; 16 missed rows) and reasoning (27 of the remaining 80 samples, i.e. 34%).

4 Ablation Study

To evaluate the significance of the two primary stages in H-STAR: (a) **Table extraction** and (b) **Adaptive reasoning**, we conducted an ablation study, removing one stage at a time. For the first condition, the final adaptive reasoning stage was substituted with a basic chain-of-thought reasoning process while maintaining the table extraction stage. Conversely, the second condition retained the adaptive reasoning stage but omitted the table extraction stage, specifically removing the row extraction and column extraction steps individually and then collectively. A detailed assessment of the contribution of each stage to the overall performance of the method is shown in Table 6. It highlights the importance of each step in the pipeline. Removing any step can result in a performance loss especially adaptive reasoning which causes a 7% drop for both TabFact and WikiTQ datasets. For a detailed error case study and further analysis, refer to the Appendix E.

Method	TabFact	WikiTQ
H-STAR	**86.51**	**68.62**
w/o row extraction	86.17	66.30
w/o column extraction	84.04	67.03
w/o table extraction	83.79	63.58
w/o adaptive reasoning	79.35	61.47

Table 6: Performance on TabFact and WikiTQ datasets.

4.1 Impact of the Hybrid Approach

We perform a quantitative study of the impact of the hybrid approach in H-STAR. Figures 16 and 17 (Appendix E) show how the use of only one of the approaches results in insufficient outputs for column and row extraction tasks, respectively. Furthermore, Table 7 shows that not employing a hybrid approach results in a significant performance loss.

Analysis. SQL methods though efficient with numerical reasoning, are highly sensitive to data variations such as irregular formatting and mixed data types, leading to a performance drop. In contrast, textual reasoning is more resilient, offering fuzzy matching and better interpretation of data. By combining textual and symbolic reasoning, H-STAR uses their complementary strengths, achieving superior performance.

Method	TabFact	WikiTQ
CoT	79.05	60.43
H-STAR $_{text}$	79.43	61.47
Text-to-SQL	52.05	42.03
H-STAR $_{sql}$	58.50	46.09
H-STAR	**86.51**	**68.62**

Table 7: Performance using only text-based/SQL-based reasoning; H-STAR $_{text}$: H-STAR with textual reasoning only; H-STAR $_{sql}$: H-STAR with symbolic reasoning only.

4.2 Symbolic vs Semantic Approaches

As shown in Table 8 removing SQL extraction drops the accuracy to 85.22% on TabFact and 64.39% on WikiTQ, and removing text extraction causes a drop to 83.74% and 60.31%, respectively. SQL reasoning removal results in 84.48% and 64.76%, but omitting text reasoning causes the largest drop, to 58.70% and 54.35%.

Method	TabFact	WikiTQ
H-STAR	**86.51**	**68.62**
w/o SQL extraction	85.22	64.39
w/o text extraction	83.74	60.31
w/o SQL reasoning	84.48	64.76
w/o text reasoning	58.70	54.35

Table 8: Performance of H-STAR after systematically removing symbolic part and semantic parts from: (1) table extraction (both row and column extraction); (2) adaptive reasoning.

Analysis. Table 8 shows that text-based approaches excel in both table extraction and reasoning. The evaluation datasets often contain noisy data that SQL-based approaches struggle with due to their reliance on structured schemas. Since text-based reasoning methods are more effective at handling such irregularities, they achieve reasonably high scores, albeit lower than our hybrid approach.

5 Key Findings

Firstly, our evaluation demonstrates that our hybrid approach achieves substantial improvements, surpassing the performance of previous state-of-the-art methods across various table reasoning tasks. Secondly, the quantitative analysis demonstrates that our 'multi-view' approach extracts tables specific to the query. Furthermore, the qualitative analysis highlights fewer errors in our table extraction method compared to prior approaches, confirming a decrease in irrelevant information.

Thirdly, our analysis indicates consistent performance even with longer tables, emphasizing our method's effectiveness in accurately extracting relevant information by filtering out noise. Lastly, the ablation study shows that decomposing the task into sub-tasks significantly enhances the overall performance, with each sub-task playing a crucial role in achieving superior results. Moreover, it also highlights the constraints of relying solely on either text or SQL approaches, which are effectively addressed by our H-STAR approach. Together, these findings emphasize the substantial advantages of our hybrid and multi-view approach in addressing complex table reasoning tasks.

6 Related Work

Table reasoning tasks require the ability to reason over unstructured queries and structured or semi-structured tables. Traditional approaches like TAPAS (Herzig et al., 2020), TAPEX (Liu et al., 2021), TABERT (Yin et al., 2020), TURL (Deng et al., 2022), PASTA (Gu et al., 2022) work on pre-training language models jointly on large-scale tabular and text data to reason in an end-to-end manner. Advancements in LLMs, allow them to learn from in-context examples, reducing inference costs. Text-to-SQL (Rajkumar et al., 2022) and Program-of-Thought (Chen et al., 2023) use symbolic methods to solve table-based tasks via Python/SQL programs. Textual-based reasoning techniques such as Table-CoT (Chen, 2023) and Tab-CoT (Ziqi and Lu, 2023) extend prompting methods such as zero- and few-shot CoT for tabular reasoning.

The decomposition of problems into smaller and manageable tasks has proven effective in solving complex reasoning challenges (Zhou et al., 2022; Khot et al., 2022). Recent techniques in table reasoning follow this approach, either by breaking tasks into fixed sub-tasks (Cheng et al., 2022; Ye et al., 2023; Nahid and Rafiei, 2024) or by employing iterative methods (Jiang et al., 2023; Zhang et al., 2023; Wang et al., 2023). BINDER (Cheng et al., 2022) is an SQL-based approach that modifies SQL statements to include LLM API calls within SQL statements. ALTER (Zhang et al., 2024a) augments both the queries along with the table data. DATER (Ye et al., 2023) and TabSQLify (Nahid and Rafiei, 2024) are table decomposition methods that use semantic and symbolic reasoning, respectively. Chain-of-Table (Wang et al., 2023) uses a textual reasoning approach to update tables iteratively before the final reasoning step. SYNTQA (Zhang et al., 2024b) is an ensemble approach employing an answer selection mechanism that selects between text-to-SQL and text-based models. The works most closely related to our approach are ReAcTable (Zhang et al., 2023) and E5 (Zhang et al., 2024c). ReAcTable extends the ReAct framework (Yao et al., 2023) to table reasoning by employing step-by-step reasoning, where it iteratively generates sub-tables using external tools like SQL and Python. Similarly, E5 introduces a multi-step process for hierarchical table question answering, interpreting table structures, generating and executing code, and performing reasoning over the results to derive answers. In contrast, H-STAR employs a fixed two-stage pipeline for table extraction and reasoning, utilizing both symbolic and semantic approaches complementarily at each stage, resulting in improved performance. A comprehensive comparison of the methodological differences and their implications for performance is provided in Appendix B.

7 Conclusion

In this study, we present H-STAR, a novel method that effectively integrates semantic and symbolic approaches, demonstrating superior performance compared to existing methods in tasks involving table reasoning. Our approach involves a two-step LLM-driven process: firstly, employing 'multi-view' table extraction to retrieve tables relevant to a query, and then, implementing adaptive reasoning to select the optimal reasoning strategy based on the input query. We address prior bottlenecks with efficient extraction and reasoning, leading to improved overall performance, particularly on longer tables. Our results highlight the need to move beyond relying solely on either technique and demonstrate the effectiveness of an integrated approach that combines the advantages of both methods. Future directions involve testing the adaptability of our methods to semi-structured, complex hierarchical, and relational tables. Enhancing the reasoning process through techniques such as self-consistency and self-verification shows promising potential.

Limitations

Our current work has primarily focused on a subset of table reasoning tasks using datasets sourced from Wikipedia. While this has laid a solid foundation, it limits exploration into diverse reasoning tasks such as table manipulation, text-to-table generation, and table augmentation, which could provide valuable insights and enhance our approach's capabilities. Additionally, our method's generalizability is confined to Wikipedia-based datasets, restricting its application to other domains that require specific domain knowledge, which our current approach lacks. Extending our approach to different domains may necessitate integrating domain-specific knowledge to ensure effective reasoning.

Furthermore, our evaluation has been limited to relatively straightforward table structures. Handling more complex data representations such as semi-structured tables, hierarchical tables, and relational databases remains unexplored territory. These structures present unique challenges that our current approach may not effectively address. Finally, our study has focused solely on the English language, potentially limiting its applicability to languages with different linguistic complexities that we have not accounted for.

Ethics Statement

We, the authors, affirm that our work adheres to the highest ethical standards in research and publication. We have carefully considered and addressed various ethical issues to ensure the responsible and fair use of computational linguistics methodologies. To facilitate reproducibility, we provide detailed information, including code, datasets (all publicly available and in compliance with their respective ethical standards), and other relevant resources. Our claims align with the experimental results, though some stochasticity is expected with black-box large language models, which we minimize by maintaining a fixed temperature. We provide comprehensive details on annotations, dataset splits, models used, and prompting methods, ensuring our work can be reliably reproduced.

Acknowledgments

This research was partly sponsored by the U.S. National Science Foundation (NSF) under Grant No. 2416728, Army Research Office under Grant Number W911NF-20-1-0080, and ONR Contract N00014-23-1-2364. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. We thank the authors of Tabsqlify and BINDER for sharing their code, and Yerram Varun for his help in setting up the implementation process. We also extend our gratitude to Jennifer Sheffield from UPenn for research support. Lastly, we appreciate the reviewing team's insightful comments.

References

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

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

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. 2019. Tabfact: A large-scale dataset for table-based fact verification. In *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, et al. 2022. Binding language models in symbolic languages. In *The Eleventh 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.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Zihui Gu, Ju Fan, Nan Tang, Preslav Nakov, Xiaoman Zhao, and Xiaoyong Du. 2022. Pasta: Table-operations aware fact verification via sentence-table cloze pre-training. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4971–4983.

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.

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.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2021. Tapex: Table pre-training via learning a neural sql executor. In *International Conference on Learning Representations*.

Tianyang Liu, Fei Wang, and Muhao Chen. 2023. Rethinking tabular data understanding with large language models. *arXiv preprint arXiv:2312.16702*.

Md Mahadi Hasan Nahid and Davood Rafiei. 2024. Tabsqlify: Enhancing reasoning capabilities of llms through table decomposition. *arXiv preprint arXiv:2404.10150*.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, et al. 2022. Fetaqa: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

R OpenAI. 2023. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2(5).

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.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models. *arXiv preprint arXiv:2204.00498*.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Ananya Singha, José Cambronero, Sumit Gulwani, Vu Le, and Chris Parnin. 2023. Tabular representation, noisy operators, and impacts on table structure understanding tasks in llms. In *NeurIPS 2023 Second Table Representation Learning Workshop*.

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*, pages 645–654.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, et al. 2023. Chain-of-table: Evolving tables in the reasoning chain for table understanding. In *The Twelfth International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

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*, pages 174–184.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426.

Han Zhang, Yuheng Ma, and Hanfang Yang. 2024a. Alter: Augmentation for large-table-based reasoning. *arXiv preprint arXiv:2407.03061*.

Siyue Zhang, Anh Tuan Luu, and Chen Zhao. 2024b. Syntqa: Synergistic table-based question answering via mixture of text-to-sql and e2e tqa. *arXiv preprint arXiv:2409.16682*.

Yunjia Zhang, Jordan Henkel, Avrilia Floratou, Joyce Cahoon, Shaleen Deep, and Jignesh M Patel. 2023. Reactable: Enhancing react for table question answering. *arXiv preprint arXiv:2310.00815*.

Zhehao Zhang, Yan Gao, and Jian-Guang Lou. 2024c. E5: Zero-shot hierarchical table analysis using augmented llms via explain, extract, execute, exhibit and extrapolate. 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 1244–1258.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

Jin Ziqi and Wei Lu. 2023. Tab-CoT: Zero-shot tabular chain of thought. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10259–10277, Toronto, Canada. Association for Computational Linguistics.

A Supplementary Analysis

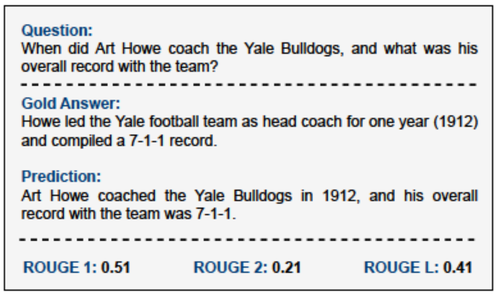
A.1 FeTaQA

The results for the FeTaQA dataset using the ROUGE scores (Lin, 2004) are given in Table 9. We observe an incremental improvement in the scores when compared to other methods on the GPT-3.5-Turbo. *The ROUGE scores focus on lexical similarities while ignoring the semantic similarity between predicted and gold outputs.* These metrics often fail to capture improvements with in-context learning, as the model doesn't learn the long-form text style from just an instruction or a few examples.

Figure 7 illustrates an example where the generated output, despite being correct in answering the question, is penalized by the *ROUGE* metric. This highlights the limitations of the metric in evaluating the correctness of the generated responses.

Prompting	ROUGE-1	ROUGE-2	ROUGE-L
PaLM-2*			
DATER	0.63	0.41	0.53
Chain-of-Table	0.66	0.44	0.56
GPT-3.5-Turbo			
Table-CoT	0.62	0.39	0.51
TabSQLify	0.58	0.35	0.48
H-STAR	0.62	0.39	0.52

Table 9: Performance on FeTaQA dataset. PaLM-2* refers to the Google proprietary PaLM-2 model.


Question:
When did Art Howe coach the Yale Bulldogs, and what was his overall record with the team?

Gold Answer:
Howe led the Yale football team as head coach for one year (1912) and compiled a 7-1-1 record.

Prediction:
Art Howe coached the Yale Bulldogs in 1912, and his overall record with the team was 7-1-1.

ROUGE 1: 0.51 ROUGE 2: 0.21 ROUGE L: 0.41

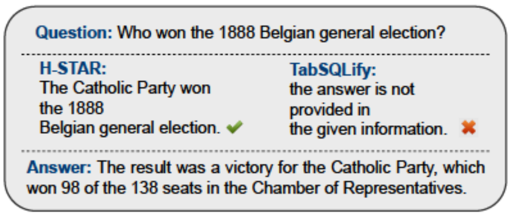
Figure 7: An example of FeTaQA representing low ROUGE scores for correct answers.

Qualitative Analysis. In this section, we perform a qualitative analysis on 100 samples of FeTaQA with TabSQLify (Nahid and Rafiei, 2024). The study aims to compare the generations of H-STAR with TabSQLify by evaluating the answer quality. The answers are evaluated based on their fluency and adequacy compared to the gold answers. Table 10 shows that H-STAR has better quality generations in 26% samples, as opposed to only 18% for TabSQLify. 'Both' indicates samples where both the algorithms perform on par. This can be attributed to the use of the same LLM. Many failures for TabSQLify are derived from an insufficient table, further highlighting the limitations of using a single-view approach for table extraction.

Method	% of samples
Both	56%
TabSQLify	18%
H-STAR	26%

Table 10: Comparison of H-STAR with TabSQLify on 100 samples.

Figure 8 presents an example in which TabSQLify returns *'information not provided'* due to an incorrect table. It can be seen that H-STAR generates an answer comparable with the ground truth for the same. Figure 9 visually depicts the outputs from both methods. It illustrates the evaluation process undertaken to compare the output quality.

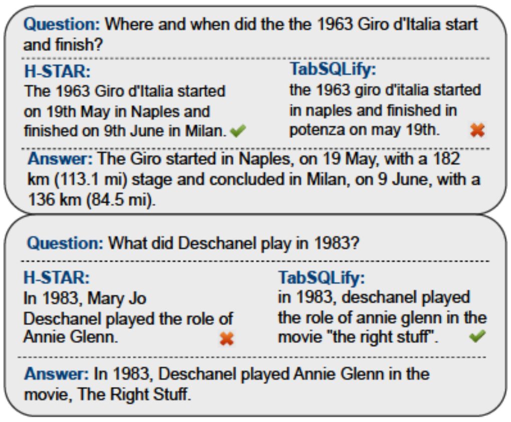

Question: Who won the 1888 Belgian general election?

H-STAR: The Catholic Party won the 1888 Belgian general election. ✓

TabSQLify: the answer is not provided in the given information. ✗

Answer: The result was a victory for the Catholic Party, which won 98 of the 138 seats in the Chamber of Representatives.

Figure 8: An example of FeTaQA dataset where TabSQLify fails as a result of insufficient table evidence.


Question: Where and when did the the 1963 Giro d'Italia start and finish?

H-STAR: The 1963 Giro d'Italia started on 19th May in Naples and finished on 9th June in Milan. ✓

TabSQLify: the 1963 giro d'italia started in naples and finished in potenza on may 19th. ✗

Answer: The Giro started in Naples, on 19 May, with a 182 km (113.1 mi) stage and concluded in Milan, on 9 June, with a 136 km (84.5 mi).

Question: What did Deschanel play in 1983?

H-STAR: In 1983, Mary Jo Deschanel played the role of Annie Glenn. ✗

TabSQLify: in 1983, deschanel played the role of annie glenn in the movie "the right stuff". ✓

Answer: In 1983, Deschanel played Annie Glenn in the movie, The Right Stuff.

Figure 9: Comparison of the outputs with TabSQLify.

Human Evaluation. We further evaluate the quality of our generations using human evaluation. We refer to (Nan et al., 2022) and evaluate our generations on four factors: **(1) Fluency** based on the nature and grammar of the output; **(2) Correctness** measuring the degree of correctness of the prediction; **(3) Adequacy** to measure whether all the necessary information is contained in the output; **(4) Faithfulness** to check whether the generation is grounded in the final extracted table used for reasoning. We ask five internal human annotators to assign a score of 1-5 for each criterion and report the percentage of samples having values of 4 or 5. Table 11 shows the results by five annotators for our method and compares it with other prior methods. The performance pattern of H-STAR is similar to human performance with comparable fluency and faithfulness while matching the trend of having higher adequacy over correctness. Although we marginally outperform other methods, there is still scope for improvement.

Method	Fluent	Correct	Adequate	Faithful
T5-large	94.6	54.8	50.4	50.4
Human	95	92.4	95.6	95.6
TableCoT	96	82	75	87
TabSQLify	97	88	84	93
H-STAR	96.6	87.6	89.6	94

Table 11: Human evaluation results on FeTaQA dataset.

A.2 Comparison With Diverse Methods

Tables 12 and 13 show results of other baseline methods on WikiTQ and TabFact datasets respectively. Our method, H-STAR, outperformed all other models by substantial margins. Despite using the smaller, less capable GPT-4o-mini, H-STAR outperforms E5 and ReAcTable which use GPT-4 on both TabFact and WikiTQ. Furthermore, H-STAR (GPT-3.5-Turbo) outperforms E5 (GPT-4) and ReAct (GPT-4) on WikiTQ.

Model	Method	Accuracy
Pre-trained/fine-tuned	TaPas	48.8
	GraPPa	52.7
	LEVER	62.9
	ITR	63.4
	SYNTQA(GPT)*	70.4
Codex[2]	GPT-3 CoT	45.7
	TableCoT	48.8
	DATER	65.9
	BINDER	61.9
	ReAcTable	65.8
GPT-3.5-Turbo	ReAcTable	52.5
	E5	50.9
	TableCoT	52.4
	StructGPT	52.2
	ALTER	67.4
	SYNTQA(GPT)*	65.2
GPT-4	ReAcTable	57.3
	E5	65.5
GPT-3.5-Turbo	**H-STAR**	**69.6**
GPT-4o-Mini		**74.9**

Table 12: Comparison of H-STAR with additional methods on WikiTQ dataset. * SYNTQA (GPT) uses GPT-3.5-Turbo to choose between SQL/text reasoning and uses: (1) fine-tuned models for reasoning; (2) GPT-3.5-Turbo for reasoning.

Model	Method	Accuracy
Pre-trained/fine-tuned	Table-BERT	68.1
	LogicFactChecker	74.3
	SAT	75.5
	TaPas	83.9
	TAPEX	85.9
	SaMoE	86.7
	PASTA	90.8
Codex	TableCoT	72.6
	DATER	85.6
	BINDER	85.1
	ReAcTable	83.1
GPT-3.5-Turbo	ReAcTable	73.1
	E5	75.6
	TableCoT	73.1
	ALTER	84.3
GPT-4	ReAcTable	83.7
	E5	88.8
GPT-3.5-Turbo	**H-STAR**	**85.0**
GPT-4o-Mini		**89.4**

Table 13: Comparison of H-STAR with additional methods on TabFact dataset.

[2]OpenAI Codex model is not available publically anymore.

A.3 Advantages of using Transposed Table

Table 14 compares the performance of H-STAR using original versus transposed tables for text-based column verification. In TabFact, H-STAR achieves 86.51% accuracy with transposed tables versus 85.22% with the original tables. On WikiTQ, accuracy is 68.62% with transposed tables, compared to 66.59% with the original tables.

Method	TabFact	WikiTQ
H-STAR (with transposed table)	86.51	68.62
H-STAR (with original table)	85.22	66.59

Table 14: Comparison between H-STAR using (a) transposed table and (b) original table for text-based column extraction.

Analysis. Transposing tables leads to better performance than using the original table for column verification. The transposed table approach is more effective for complex datasets such as WikiTQ, which are harder to reason on and contain longer tables.

B Comparison with Other Methods

While H-STAR shares similarities with ReAct and E5 in combining textual reasoning with tool-based methods, there are key differences. Like E5, H-STAR employs table extraction to filter irrelevant data and leverages LLMs' code generation capabilities for table reasoning. However, there are significant differences.

1. Integration of Symbolic and Semantic Reasoning. H-STAR combines SQL-based and text-based reasoning, while ReAct merges textual reasoning with actions. While ReAct uses tools primarily for actions and E5 applies code generation only for table extraction, both rely solely on semantic approaches for reasoning. In contrast, H-STAR applies SQL for quantitative tasks and text for lexical queries.

2. Fixed Multi-Stage Approach. ReAct prompts LLMs to alternate between reasoning and actions, allowing interaction with external sources. H-STAR, on the other hand, splits tabular reasoning into two stages: (1) Table Extraction, and (2) Adaptive Reasoning, blending SQL and text reasoning. This hybrid approach sets H-STAR apart from ReAct's focus on textual reasoning and interaction with the environment.

3. Table Extraction Approach. H-STAR distinguishes itself by combining symbolic and semantic methods for table extraction, rather than relying exclusively on either approach. Furthermore, H-STAR uses a 'multi-view' approach for table extraction, identifying relevant columns from both original and transposed tables before selecting rows, reducing irrelevant data, and improving focus. In contrast, ReAct does not employ any extraction method.

4. Adaptive Reasoning Strategy. Unlike E5, H-STAR uses SQL to support semantic reasoning for quantitative tasks while relying solely on semantic reasoning for lexical queries. In contrast, ReAct interleaves reasoning and actions without adapting to task type. H-STAR's hybrid approach, combining SQL logic and text understanding, excels at handling diverse tasks, outperforming ReAct on table reasoning tasks.

C Input Table Format

The input table format changes depending on the type of reasoning used. Since H-STAR uses a hybrid approach, the input table format also varies depending on the type of reasoning used. Besides the question and the in-context examples, the table is accompanied by a list of columns and a table caption if available. Providing more context aids in better semantic understanding (Singha et al., 2023; Sui et al., 2024).

C.1 Textual Reasoning

We convert the tables into linear, sequential text. Continuing with the format of previous textual reasoning methods (Ye et al., 2023; Chen, 2023; Wang et al., 2023), we adopt the PIPE encoding i.e. plain text separated by '|'. Furthermore, we append the table with the caption and the list of columns similar to the format in Section C.2. Example of the input table:

```
table caption:  2012-13 Exeter City
F.C. season
/
col :  name | league | total
row 1:  danny coles | 3 | 3
row 4:  john o'flynn | 11 | 12
row 8:  jamie cureton | 20 | 20
*/
columns:  ['name', 'league', 'total']
```

C.2 Symbolic Reasoning

We adopt the table prompt format from previous SQL-based methods Text-to-SQL (Rajkumar et al., 2022; Cheng et al., 2022; Nahid and Rafiei, 2024). We include (1) the table schema containing `CREATE TABLE` followed by the schema, (2) the table header and all the rows separated by tabs, and (3) the list of columns along with the corresponding query. If the prompt exceeds the context limit, we truncate the table rows to fit within the limit. Example of the input table:

```
CREATE TABLE 2012-13 Exeter City F.C.
season(
  row_id int,
  name text,
  league int,
  total int)
/
All rows of the table:
SELECT * FROM w;
row_id   name    league   total
1   danny coles   3   3
4   john o'flynn   11   12
8   jamie cureton   20   20
/
columns:  ['name', 'league', 'total']
```

D Implementation Details

We provide the implementation details including the hyperparameters and prompt details for each of the individual steps. H-STAR operates in two key stages: (1) Table Extraction and (2) Adaptive Reasoning. The Table Extraction phase includes column extraction (SQL-based: $\texttt{col}_{\texttt{sql}}$ and text-based: $\texttt{col}_{\texttt{text}}$) and row extraction (SQL-based: $\texttt{row}_{\texttt{sql}}$ and text-based: $\texttt{row}_{\texttt{text}}$). The Adaptive Reasoning phase consists of SQL-based ($\texttt{f}_{\texttt{sql}}$) and text-based ($\texttt{f}_{\texttt{text}}$) reasoning components.

D.1 Input Prompts

Figures 10, 11, 12, 13, 14 and 15 illustrate the input prompts containing sample in-context learning examples for the steps $\texttt{col}_{\texttt{sql}}$, $\texttt{col}_{\texttt{text}}$, $\texttt{row}_{\texttt{sql}}$, $\texttt{row}_{\texttt{text}}$, $\texttt{f}_{\texttt{sql}}$, and $\texttt{f}_{\texttt{text}}$ respectively.

D.2 Hyperparameters

Table 15 provides the details of PaLM-2 hyperparameters used for the WikiTQ and TabFact datasets. Table 16 showcases the hyperparameter setting for GPT-3.5-Turbo. The hyperparameters 'samples' indicate the number of outputs taken for each step of the pipeline whereas the 'examples' indicate the number of few-shot demonstrations for each step.

Function	temperature	top_p	output_tokens	samples	examples
			WikiTQ		
$\texttt{col}_{\texttt{sql}}$	0.4	1.0	512	2	3
$\texttt{col}_{\texttt{text}}$	0.7	1.0	512	2	3
$\texttt{row}_{\texttt{sql}}$	0.4	1.0	512	2	3
$\texttt{row}_{\texttt{text}}$	0.7	1.0	512	2	2
$\texttt{f}_{\texttt{sql}}$	0.1	1.0	512	1	3
$\texttt{f}_{\texttt{text}}$	0.0	1.0	512	1	4
			TabFact		
$\texttt{col}_{\texttt{sql}}$	0.4	1.0	512	2	4
$\texttt{col}_{\texttt{text}}$	0.7	1.0	512	2	3
$\texttt{row}_{\texttt{sql}}$	0.4	1.0	512	2	4
$\texttt{row}_{\texttt{text}}$	0.7	1.0	512	2	3
$\texttt{f}_{\texttt{sql}}$	0.1	1.0	512	1	3
$\texttt{f}_{\texttt{text}}$	0.0	1.0	512	1	5

Table 15: Hyperparameter settings for PaLM-2.

Function	temperature	top_p	output_tokens	samples	examples
			WikiTQ		
$\texttt{col}_{\texttt{sql}}$	0.3	1.0	512	2	3
$\texttt{col}_{\texttt{text}}$	0.4	1.0	512	2	2
$\texttt{row}_{\texttt{sql}}$	0.3	1.0	512	2	3
$\texttt{row}_{\texttt{text}}$	0.4	1.0	512	2	3
$\texttt{f}_{\texttt{sql}}$	0.1	1.0	512	1	4
$\texttt{f}_{\texttt{text}}$	0.0	1.0	512	1	4
			TabFact		
$\texttt{col}_{\texttt{sql}}$	0.2	1.0	512	2	4
$\texttt{col}_{\texttt{text}}$	0.4	1.0	512	2	3
$\texttt{row}_{\texttt{sql}}$	0.4	1.0	512	2	4
$\texttt{row}_{\texttt{text}}$	0.5	1.0	512	2	3
$\texttt{f}_{\texttt{sql}}$	0.1	1.0	512	1	4
$\texttt{f}_{\texttt{text}}$	0.0	1.0	512	1	5

Table 16: Hyperparameter settings for GPT-3.5-Turbo.

E Error Analysis: Case Study

Figures 16 and 17 showcase how semantic and symbolic approaches can individually fail for the column and row extraction steps. Figures 18, 19, 20 and 21 illustrate error cases for each individual step within H-STAR.

SQL-based Column Extraction

You are given the table schema, and the table along with the corresponding statement. Write a simple SQLite program for selecting the required columns only, to help answer the question correctly. The SQLite program need not directly answer the question. Assume you always have enough information when executing the SQLite. Fuzzy match data if unsure.

```
1. Plan:
 - Identify critical values and ranges from the table related to the statement.
 - Make use of your domain knowledge to find the correct approach to solve the question.
 - Always select the column with special aggregate values like 'total'.
2. Retrieval:
 - Generate a simple SQL program extracting the relevant columns
 - SQL: SELECT <COLUMNS> FROM w;
 - Evidence: f_col([column names])

Response Format:
Begin your response with 'Output' and include:
- Plan: Write the plan for column extraction along with a reasoning chain
- Retrieval: Write a simple SQL query
 Evidence: f_col([column names])
```

Before you return the answer, review your outputs and make sure you have followed all the given instructions. Make sure to only select columns.

```
<input>
CREATE TABLE Płock Governorate(
    row_id int,
    language text,
    number int,
    percentage (%) text,
    males int,
    females int)
/*
All rows of the table:
SELECT * FROM w;
row_id   language   number   percentage (%)   males   females
0   polish   447685   80.86   216794   230891
1   yiddish   51215   9.25   24538   26677
2   german   35931   6.49   17409   18522
7   total   553633   100   275652   277981
*/
columns: ['row_id', 'language', 'number', 'percentage (%)', 'males', 'females']
Q: how many male and female german speakers are there?
<output>
1. Plan:
 - We need to count the number of male and female speakers
 - The column 'language' with the aggregate value 'total' must be included
 - We then select the 'males', 'females', 'language'
2. Retrieval:
 # SQL format: SELECT <column names> FROM w;
 - SQL: SELECT `language`, `males`, `females` FROM w;
   # Only output the column names from the column list
 - Evidence: f_col([language, males, females])
```

Figure 10: Prompt for SQL-based column extraction ($\texttt{col}_{\texttt{sql}}$).

Text-based Column Extraction

You need to select all the relevant columns from the table and obtain a relevant sub-table to the question. The sub-table should then be used to answer the question. Please ensure that your final answer matches your reasoning. Your task is to perform the following actions

1. Plan:
   Identify critical values and ranges from the table related to the question.
2. Retrieval:
   Identify all table columns containing question-relevant information. Explain the relevance of each selected piece of evidence.

Response Format: Strictly follow the given format only.
Begin your response with 'Output' and include:
- Plan: Write the plan for column extraction along with a reasoning chain
- Retrieval: Based on the plan, identify the columns followed by a reasoning chain
  Evidence: f_col([column names])

```
<input>
/*
row: row 1 | row 2 | row 3 | row 4 | row 5
pick: 1 | 2 | 3 | 4 | 5
player: jay washington | alex cabagnot | dennis miranda | ato ular | kameron vales
country of origin: united states | united states | philippines | philippines | united states
pba team: air21 express | sta lucia realtors | coca-cola tigers | blackwater bossing | converge fiberxers
college: eckerd | hawaii - hilo | feu | mpbl | ateneo
*/
Q: Which college has most players belonging to asian countries?
<output>
1. Plan:
 a. Extract data from the 'country of origin' column to create a new column
 b. The table columns 'college', and 'player' contain the question-specific information
2. Retrieval:
Explanation: Look at the 'college', and 'player' columns to identify unique colleges and player names. For each row, extract the ethnicity from the 'country of origin' column to create a new column to identify Asian countries.
Evidence: f_col([player, country of origin, college])
```

Figure 11: Prompt for Column-based column verification ($\texttt{col}_{\texttt{text}}$).

SQL-based Row Extraction

~~~~
You are given the table schema, and the table along with the corresponding question. Your task is only to write an SQLite program to create a subtable to help answer the question. You must not answer the question but you must only find the relevant suitable. Write a simple but correct SQLite program.

1. Decompose:
  Find all the subtasks from the main question.
2. Retrieval:
  Generate an SQLite program for each subtask and then combine using only an 'OR' clause.

Response Format:
Begin your response with 'Output: ' and include all the following components:
 - Decompose: Divide the question into smaller manageable sub-tasks; one task for each condition
 - Retrieval: A basic SQLite program for each condition followed by a combined SQLite

**NOTE**
- Use the column names as it is as given in the column list.
- Use 'OR' to combine conditions.
- The SQLite program must be only of the format: SELECT <column names> FROM w WHERE <conditions>;

Be very careful, think, and ensure that all the above instructions are followed, and only then return your response.
```
<input>
CREATE TABLE British_Records_in_Athletics(
        row_id int,
        event text,
        place text);

/*
3 example rows:
SELECT * FROM British_Records_in_Athletics LIMIT 3;
row_id      event       place
0       5 km søfteland , norway
1       5 miles      york , united kingdom
2       10 km        burrator , united kingdom
*/
columns: ['row_id', 'event', 'place']
Q: how many events took place within the united kingdom?
<output>
1. Decompose:
 - Part 1: Find all instances of events that took place within the united kingdom
 # Do not use COUNT() clause
 # The SQLite program must be only of the format: SELECT <column names> FROM w WHERE <conditions>;
  SELECT event, place FROM w WHERE place LIKE '%united kingdom%';
2. Retrieval:
  SQL: SELECT event, place FROM w WHERE place LIKE '%united kingdom%';

```
~~~~

Figure 12: Prompt for SQL-based row selection ($\texttt{row}_{\texttt{sql}}$).

Text-based Row Extraction

~~~~
You are provided an initial response containing a response, a table, and a question. Your task is to give feedback about the initial response and find the omitted rows if any.

1. Assessment:
   - Assess the question
2. Feedback: Based on the nature of the question, provide the feedback.
   a. Calculation/comparison-based question:
     - Check if the comparison is limited to some rows or if it consists of the entire table
   b. Order/temporal-based question:
     - Check if the response includes the row containing the mentioned entity
     - Check if the response contains relative rows mentioned in the question
   c. Entity-specific question:
     - Check if any row about the entity in the question are not included
Add rows if not present in the response.
Rows: f_row([rows])

Response Format:
Begin your response with 'Output' and must always include the following:
 - Assessment: Assess the question and the response
 - Feedback: Provide a step-by step feedback on the suggested answers
   Rows: f_row([rows])

Follow all instructions given. Check if all instructions have been followed before returning the answer.
```
<input>
/*
col : rank | athlete
row 0 : 1 | Manjeet Kaur
row 1 : 2 | Olga Tereshkova
row 2 : 3 | Pinki Pramanik
row 3 : 4 | Tang Xiaoyin
row 4 : 5 | Marina Maslyonko
*/
Q: tell me who finished immediately after manjeet kaur
Initial Response: ['row 1']
<output>
1. Assessment:
  - We are asked to find the athlete after manjeet kaur.
  - Thus, we need to return the row for manjeet kaur and the row after it.
2. Feedback:
  - Is the row for Manjeet Kaur included in the initial response? A: No; add(row 0)
  - What is the row after Manjeet Kaur (row 0)? A: row 1; add(row 1)
  - Is the row after Manjeet Kaur included in the initial response? A: Yes
Rows: f_row([row 0, row 1])

```
~~~~

Figure 13: Prompt for text-based row verification ($\texttt{row}_{\texttt{text}}$).

SQL-based Reasoning

~~~~
You must carefully understand the table and the corresponding question. Your task is to determine whether
you need an SQLite program to solve the question. The final SQLite program must be simple and must be only
a single program.
Always remember, that most questions do not need an SQLite program, and you must simply return 'None'.

1. Nature of the Question:
Evaluate the nature of the question. For tasks involving computation, counting, or if the question involves
column selection for extensive tables, the solution is again 'SQL'
Make sure to use the SQLite program for mathematical questions.

2. Final Output:
A. Based on your analysis, if the solution is 'SQL', then you must write an SQLite program to extract a
subtable to answer the question. The SQLite does not need to directly answer the question. Generate an
SQLite that is less complex but ensure that it is correct. Try to use fuzzy-match for values if you are not sure
about the values. Generate a step-by-step SQLite program to answer the question.
B. If the solution is not SQL, then simply return 'None'

Response Format:
Begin your response with 'Output: ' and always include the following:
  - Nature of the Question: Evaluation based on the nature of the question.
  - Final Output: 'None' if SQL is not required otherwise the SQLite program is the solution.

Check if all instructions have been followed before returning the answer.
```

<input>
CREATE TABLE 2008 women 's british open(
        row_id int,
        player text,
        player country text)
/*
3 example rows:
SELECT * FROM w LIMIT 3;
row_id      player      player_country
0       juli inkster united states
1       momoko ueda     japan
2       laura diaz  united states
*/
columns: ['row_id', 'player', 'player country']
statement: count the number of players from united states and japan
<output>
1. Nature of the Question:
  - This question involves counting the number of players from a specific countries.
  - Counting data points within a table is a task suited for SQL.
2. Final Output: Since the task is SQL,
  - Step 1: Group players by country and count the occurrences
    SELECT `player country`, COUNT(*) AS total_players FROM w GROUP BY `player country`;
SQL: SELECT `player country`, COUNT(*) AS total_players_per_country FROM w GROUP BY `player
country`;

```
~~~~

Figure 14: Prompt for SQL-based reasoning ($\texttt{f}_{\texttt{sql}}$).

Text-based Reasoning

~~~~
You are an expert on table data. You must use the table data and the additional evidence to answer the given question.

- Divide the main statement into sub-tasks and answer each sub-task
- Based on the answers, check whether the statement is supported by the table

**NOTE**
Please be extremely careful, pause, make sure all instructions have been followed, and only then output the answer

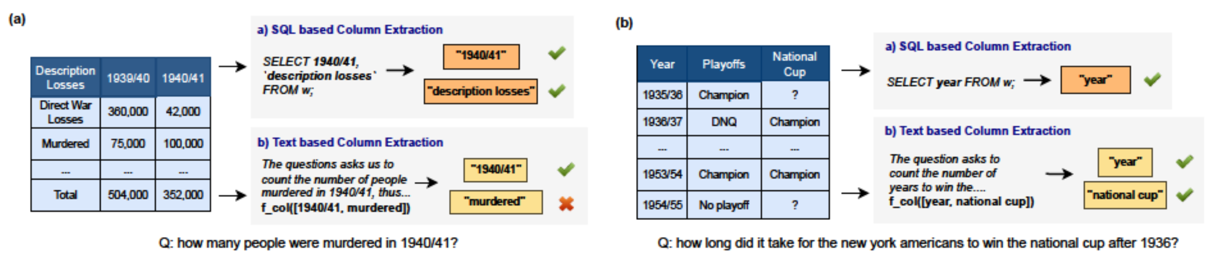
Response Format:
Begin your response with 'Output: ' and always include the following:
- Decompose: Divide the main question into sub-tasks and answer each sub-task
- Final Answer: Strictly output as a short phrase starting with `therefore, the answer is: "AnswerName1", "AnswerName2"...` form, no other form

- Read the question carefully, understand, and return what the question asks.
- Be careful, make sure you have followed all instructions, and only then return the output.
```
<input>
/*
col : rank | cyclist | team
row 0 : alejandro valverde (esp) | caisse d'epargne
row 1 : alexandr kolobnev (rus) | team csc saxo bank
row 2 : davide rebellin (ita) | gerolsteiner
row 3 : paolo bettini (ita) | quick step
row 4 : franco pellizotti (ita) | liquigas
row 5 : denis menchov (rus) | rabobank
row 6 : samuel sánchez (esp) | euskaltel-euskadi
row 7 : stéphane goubert (fra) | ag2r-la mondiale
row 8 : haimar zubeldia (esp) | euskaltel-euskadi
row 9 : david moncoutié (fra) | cofidis
*/
columns: ['rank', 'cyclist', 'team']
Q: which country had the most cyclists finish within the top 10?
<output>

Here is an additional evidence to help the answering process.
Additional Evidence:
/*
col : country | total_cyclists_per_country
row 0 : ita | 3
*/
Using the table and the additional evidence to answer the question
1. Decompose:
   - #1: Number of cyclists from Italy in the top 10 = 3
2. Final Answer: Therefore, the answer is: "Italy"

```
~~~~

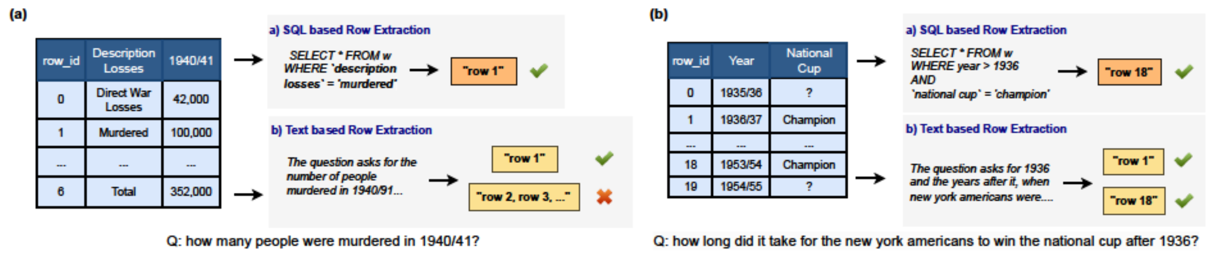
Figure 15: Prompt for text-based reasoning ($\mathtt{f}_{\mathtt{text}}$).


(a)
a) SQL based Column Extraction
SELECT 1940/41, `description losses` FROM w;
"1940/41"
"description losses"
b) Text based Column Extraction
The questions asks us to count the number of people murdered in 1940/41, thus... f_col([1940/41, murdered])
"1940/41"
"murdered"
Q: how many people were murdered in 1940/41?
(b)
a) SQL based Column Extraction
SELECT year FROM w;
"year"
b) Text based Column Extraction
The question asks to count the number of years to win the.... f_col([year, national cup])
"year"
"national cup"
Q: how long did it take for the new york americans to win the national cup after 1936?

Description Losses	1939/40	1940/41
Direct War Losses	360,000	42,000
Murdered	75,000	100,000
...	...	...
Total	504,000	352,000

Year	Playoffs	National Cup
1935/36	Champion	?
1936/37	DNQ	Champion
...	...	...
1953/54	Champion	Champion
1954/55	No playoff	?

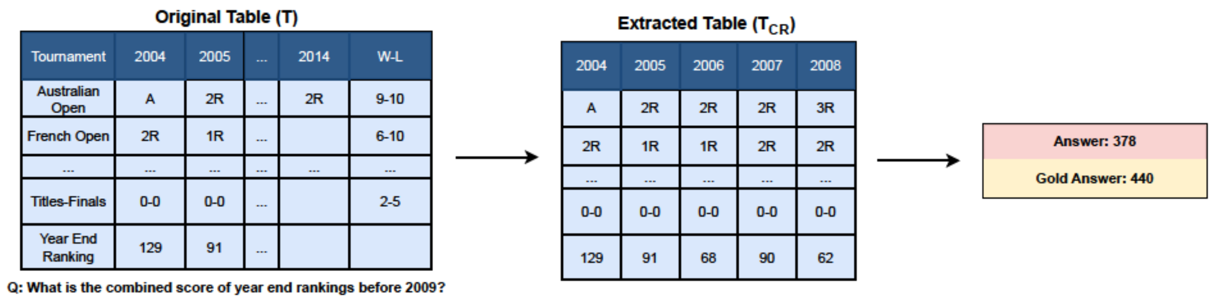
Figure 16: Comparison of SQL-based and text-based column extraction methods for answering table queries: (a) SQL-based selection accurately identifies the columns ("1940/41", "description losses"), while text-based selection mistakenly includes "murdered" as a table column. (b) The text-based method selects both required columns ("year" and "national cup"), while the SQL-based approach overlooks "national cup" as a requirement.


(a)
a) SQL based Row Extraction
SELECT * FROM w WHERE `description losses` = 'murdered'
"row 1"
b) Text based Row Extraction
The question asks for the number of people murdered in 1940/91...
"row 1"
"row 2, row 3, ..."
Q: how many people were murdered in 1940/41?
(b)
a) SQL based Row Extraction
SELECT * FROM w WHERE year > 1936 AND `national cup` = 'champion'
"row 18"
b) Text based Row Extraction
The question asks for 1936 and the years after it, when new york americans were....
"row 1"
"row 18"
Q: how long did it take for the new york americans to win the national cup after 1936?

row_id	Description Losses	1940/41
0	Direct War Losses	42,000
1	Murdered	100,000
...	...	...
6	Total	352,000

row_id	Year	National Cup
0	1935/36	?
1	1936/37	Champion
...	...	...
18	1953/54	Champion
19	1954/55	?

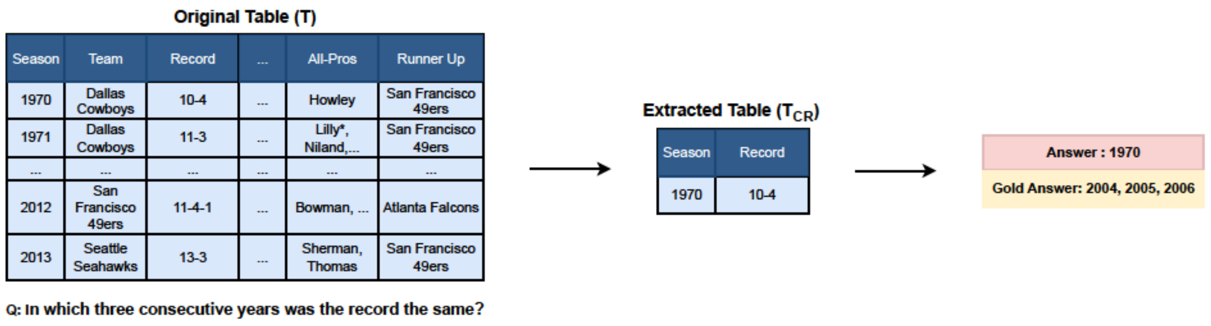
Figure 17: Comparison of SQL-based and text-based row extraction methods for answering table queries. (a) SQL-based selection accurately selects the necessary rows ("Murdered"), while text-based selection incorrectly includes additional rows; (b) Text-based method correctly selects relevant rows ("1953/54", "1936/37"), while SQL-based selection misinterprets the query, including only the row for the year "1953/54".


Original Table (T)
Extracted Table (T_{CR})
Answer: 378
Gold Answer: 440
Q: What is the combined score of year end rankings before 2009?

Tournament	2004	2005	...	2014	W-L
Australian Open	A	2R	...	2R	9-10
French Open	2R	1R	...		6-10
...	...	...	...	...	...
Titles-Finals	0-0	0-0	...		2-5
Year End Ranking	129	91	...		

2004	2005	2006	2007	2008
A	2R	2R	2R	3R
2R	1R	1R	2R	2R
...	...	...	...	...
0-0	0-0	0-0	0-0	0-0
129	91	68	90	62

Figure 18: **Case Study: 'Missing Columns'.** An illustration of the error 'Missing Columns'. In the above example, the extracted table does not contain the column 'Tournament' that contains the 'Year End Ranking'. The omission of the 'Tournament' column affects the downstream steps in the pipeline leading to an insufficient row extraction and reasoning.



Original Table (T)

Season	Team	Record	...	All-Pros	Runner Up
1970	Dallas Cowboys	10-4	...	Howley	San Francisco 49ers
1971	Dallas Cowboys	11-3	...	Lilly*, Niland,...	San Francisco 49ers
...	...	...	...	...	...
2012	San Francisco 49ers	11-4-1	...	Bowman, ...	Atlanta Falcons
2013	Seattle Seahawks	13-3	...	Sherman, Thomas	San Francisco 49ers

Q: In which three consecutive years was the record the same?

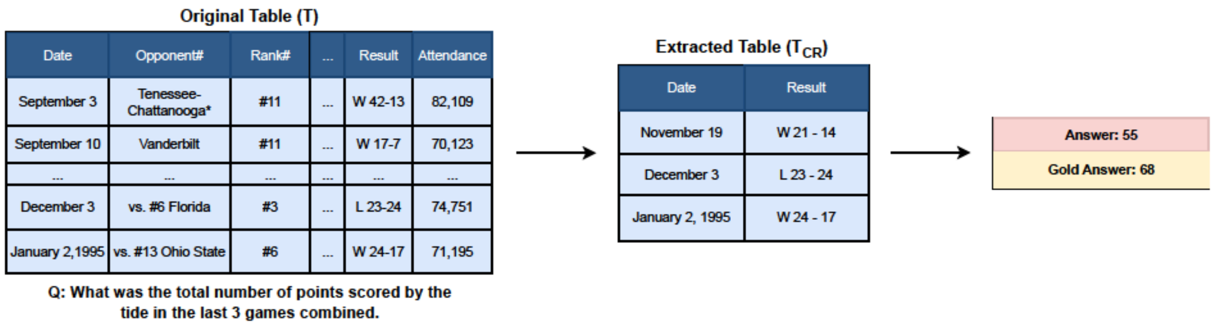
Extracted Table (T_{CR})

Season	Record
1970	10-4

Answer : 1970

Gold Answer: 2004, 2005, 2006

Figure 19: **Case Study: 'Missing Rows'.** An illustration of the error 'Missing Rows'. From the above example, it can be seen that while the extracted columns are correct, the row extraction is incorrect. Thus, as the extracted table lacks the necessary information, the final answer is incorrect.



Original Table (T)

Date	Opponent#	Rank#	...	Result	Attendance
September 3	Tenessee-Chattanooga*	#11	...	W 42-13	82,109
September 10	Vanderbilt	#11	...	W 17-7	70,123
...	...	...	...	...	...
December 3	vs. #6 Florida	#3	...	L 23-24	74,751
January 2,1995	vs. #13 Ohio State	#6	...	W 24-17	71,195

Q: What was the total number of points scored by the tide in the last 3 games combined.

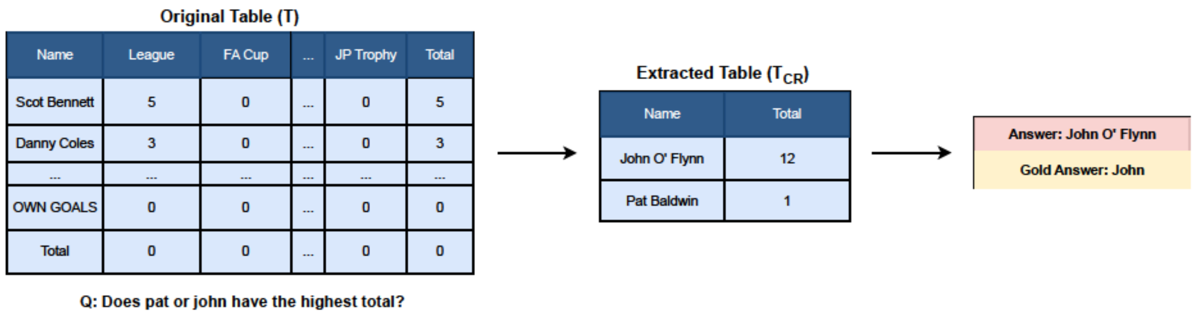
Extracted Table (T_{CR})

Date	Result
November 19	W 21 - 14
December 3	L 23 - 24
January 2, 1995	W 24 - 17

Answer: 55

Gold Answer: 68

Figure 20: **Case Study: 'Incorrect Reasoning'.** An illustration of the error 'Incorrect Reasoning'. Despite the correct question-relevant table being extracted, the model misinterprets the question. It adds up the scores by the opposing team instead thus returning an incorrect answer.



Original Table (T)

Name	League	FA Cup	...	JP Trophy	Total
Scot Bennett	5	0	...	0	5
Danny Coles	3	0	...	0	3
...	...	...	...	...	...
OWN GOALS	0	0	...	0	0
Total	0	0	...	0	0

Q: Does pat or john have the highest total?

Extracted Table (T_{CR})

Name	Total
John O' Flynn	12
Pat Baldwin	1

Answer: John O' Flynn

Gold Answer: John

Figure 21: **Case Study: 'Incorrect Annotation'.** An illustration of the error 'Incorrect Annotation'. In the example, the query-specific table is correctly extracted and the final reasoning leads to the correct answer. However, the prediction is penalized for not being an 'exact match'.