# No Universal Prompt: Unifying Reasoning through Adaptive Prompting for Temporal Table Reasoning

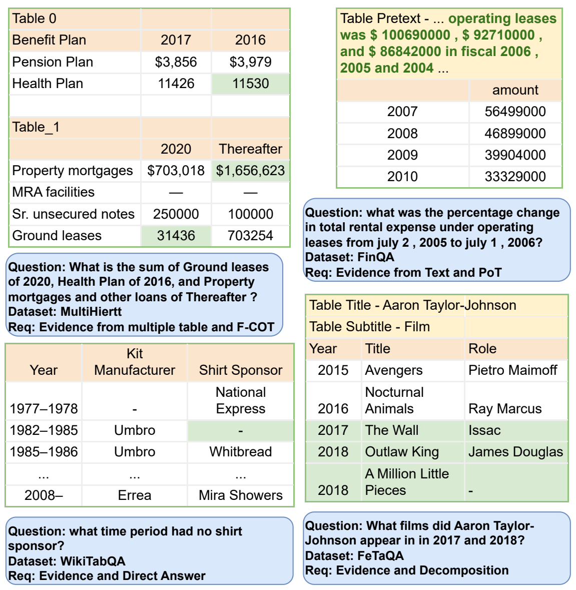
**Abhishek Rajgaria**[1*], **Kushagra Dixit**[1*†], **Mayank Vyas**[2], **Harshavardhan Kalalbandi**[3†]
**Dan Roth**[4], **Vivek Gupta**[2‡]
[1]University of Utah, [2]Arizona State University, [3] University of California Riverside,
[4]University of Pennsylvania
abhishek.rajgaria@utah.edu, kushagra.dixit@utah.edu, mvyas7@asu.edu. hkala002@ucr.edu
danroth@seas.upenn.edu, vgupt140@asu.edu

## Abstract

Temporal Table Reasoning poses a significant challenge for Large Language Models (LLMs), requiring effective reasoning to extract relevant insights. Despite existence of multiple prompting methods, their impact on table reasoning remains largely unexplored. Furthermore, model performance varies drastically across different table and context structures, making it difficult to determine an optimal approach. This work investigates multiple prompting technique on diverse table types to determine that performance depends on factors such as *entity type, table structure, requirement of additional context and question complexity*, with *"NO"* single method consistently outperforming others. To address this, we introduce SEAR, an *adaptive prompting* framework inspired by human reasoning that dynamically adjusts to context and integrates structured reasoning. SEAR_Unified, its cost-efficient variant. We also demonstrate that optional table refactoring (preprocessing) enhances both approaches when tables lack structural consistency. Our results demonstrate that SEAR prompts achieve superior performance across all table types compared to baseline prompting techniques.

## 1 Introduction

Temporal table reasoning presents a unique challenge, requiring Large Language Models (LLMs) to interpret tabular data while capturing embedded temporal relationships. Unlike static tables that provide a fixed snapshot of information, temporal tables evolve over time, incorporating event sequences, timestamps, and dynamic updates. Reasoning over such structures is essential for tasks like financial forecasting, historical trend analysis, medical diagnosis, and event-based decision making (Gupta et al., 2023; Xiong et al., 2024). However, existing LLMs often struggle to model these intricate temporal dependencies, underscoring the need for more effective reasoning frameworks.



| Table 0 | | |
|---|---|---|
| Benefit Plan | 2017 | 2016 |
| Pension Plan | $3,856 | $3,979 |
| Health Plan | 11426 | 11530 |
| | | |
| Table_1 | | |
| | 2020 | Thereafter |
| Property mortgages | $703,018 | $1,656,623 |
| MRA facilities | — | — |
| Sr. unsecured notes | 250000 | 100000 |
| Ground leases | 31436 | 703254 |

**Question: What is the sum of Ground leases of 2020, Health Plan of 2016, and Property mortgages and other loans of Thereafter ?**
**Dataset: MultiHiertt**
**Req: Evidence from multiple table and F-COT**

| Year | Kit Manufacturer | Shirt Sponsor |
|---|---|---|
| 1977–1978 | - | National Express |
| 1982–1985 | Umbro | - |
| 1985–1986 | Umbro | Whitbread |
| ... | ... | ... |
| 2008– | Errea | Mira Showers |

**Question: what time period had no shirt sponsor?**
**Dataset: WikiTabQA**
**Req: Evidence and Direct Answer**

Table Pretext - ... **operating leases was $ 100690000 , $ 92710000 , and $ 86842000 in fiscal 2006 , 2005 and 2004** ...

| | amount |
|---|---|
| 2007 | 56499000 |
| 2008 | 46899000 |
| 2009 | 39904000 |
| 2010 | 33329000 |

**Question: what was the percentage change in total rental expense under operating leases from july 2 , 2005 to july 1 , 2006?**
**Dataset: FinQA**
**Req: Evidence from Text and PoT**

| Table Title - Aaron Taylor-Johnson | | |
|---|---|---|
| Table Subtitle - Film | | |
| Year | Title | Role |
| 2015 | Avengers | Pietro Maimoff |
| 2016 | Nocturnal Animals | Ray Marcus |
| 2017 | The Wall | Issac |
| 2018 | Outlaw King | James Douglas |
| 2018 | A Million Little Pieces | - |

**Question: What films did Aaron Taylor-Johnson appear in in 2017 and 2018?**
**Dataset: FeTaQA**
**Req: Evidence and Decomposition**

Figure 1: Examples of Different Table and Contextual structure, taken from different datasets with efficient reasoning method based on specific question.

Recent work has demonstrated that LLMs can improve table reasoning performance through advanced prompting strategies (Zhang et al., 2025). Nevertheless, studies such as Wang and Zhao (2024) highlight persistent challenges in temporal reasoning, with models often failing to track evolving data or infer event sequences reliably. Moreover, most existing approaches rely on single-step prompting methods such as direct prompting or chain-of-thought reasoning (Wei et al., 2022) which frequently fail to generalize across diverse table structures and time-sensitive queries.

Although several prompting techniques have been proposed to improve LLM reasoning, their effectiveness for temporal table reasoning remains

[*]These authors contributed equally.
[†]Work done during internship at UPenn.
[‡]primary mentor corresponding author.

underexplored. In this study, we evaluate five single-step prompting methods as baselines Chain-of-Thought (CoT), Evidence Extraction, Decomposition, Faithful CoT (Radhakrishnan et al., 2023), and Program of Thought (PoT) (Chen et al., 2023). Each baseline aims to enhance logical and numerical reasoning, yet their impact on temporal table reasoning performance has not been systematically analyzed. Furthermore, we evaluate an extensive set of baselines covering structural, temporal, and agentic reasoning approaches.

This study addresses this gap by analyzing the performance of multiple established baselines and a novel adaptive reasoning strategy on a Temporal Tabular Question Answering (TTQA) task. We aim to answer the following research questions: $(RQ1)$ Given a table and a question, which reasoning strategy should be employed?, $(RQ2)$ Is there a Single reasoning method that can perform well across all types of tabular structure?, and $(RQ3)$ Is there a unified representation that can encapsulate all different tabular structures in most effective manner for the TTQA task?

To address these research questions, we conducted experiments on eight distinct tabular structures using multiple state-of-the-art LLMs for the TTQA task. To overcome the limitations of existing baselines we propose **SEAR** (Select-Elaborate-Answer & Reasoning) framework, a novel adaptive prompting strategy. Our motivation can be likened to a carpenter building a chair. They have many tools, such as a hammer, saw and drill. Each is capable of performing part of the task, but none of them can build the whole chair. It is the skillful selection and combination of these tools that brings the chair to life. Similarly, SEAR equips models with multiple reasoning tools, and then it is on the model's capability to choose them for solving the task at hand. From Table 11, we observe that models actually use multiple tools to answer these questions.

SEAR operates in three distinct phases. In the Initial **Select** phase, it identifies high-level crucial steps, in the subsequent **Elaborate** phase it refines these steps by adding detailed instructions, ensuring comprehensive road map. Finally, the **Answer & Reasoning** phase leverages the structured plan to deliver accurate answers, supported by clean, logical explanations and where necessary, includes integration of Python code for computational tasks.

Furthermore, we combined these three phases to create a single step reasoning strategy, which we call SEAR_Unified. Our results show that SEAR_Unified outperforms all single step baseline reasoning strategies by significant margins, and even standard 3-step SEAR and existing multi-step reasoning strategies such as Self Discover (Zhou et al., 2024). This demonstrates the supremacy and efficacy of our proposed reasoning strategy. Additionally, our study also includes detailed analysis of refactoring process, wherein we transform diverse tabular structure into a unified representation ("Refactor"), Our main contributions are:

- ***Benchmarking Prompting Methods***: We evaluate five single-step prompting methods and show that their effectiveness varies based on table structure, entity type, sparsity and question complexity.
- ***Adaptive Reasoning Framework***: We introduce SEAR, a multi-step adaptive prompting approach that generalizes well across diverse table structures, also we integrate them into a single unified adaptive prompt SEAR_Unified, outperforming individual methods.
- ***Table Structure Refactoring***: We propose refactoring as an enhancement, demonstrating its effectiveness in improving model reasoning by optimized table representation.
- ***Comprehensive Evaluation***: We conduct a systematic analysis across various table types, highlighting the impact of different reasoning strategies and structure modifications.

Our dataset, along with all necessary code scripts, is available at `https://coral-lab-asu.github.io/SEAR`

## 2 Why is Temporal Table Reasoning Challenging?

Temporal table QA requires models to reason over structured data while accounting for time-dependent relationships. This challenge arises from three key factors: the diverse structures of tables, the domain-specific reasoning requirements, and the complexity of the questions asked.

**Structural Variability.** Tables range from simple grids to hierarchical or semi-structured layouts with merged cells and implicit links (e.g., HiTab's multi-level indexes, HybridQA's tables mixed with text). They also come in diverse file formats (CSV, HTML, Markdown), so parsing must be flexible. SEAR first flattens and standardises these varied structures, making them easier for downstream reasoning.

**Domain-Specific Complexity.** Reasoning strategies must adapt to the table's domain. Wikipedia-based datasets like WikiTableQuestions demand general factual reasoning and entity linking. Financial datasets like FinQA or TAT-QA emphasize numerical reasoning, requiring multi-step arithmetic and temporal trend analysis. SEAR dynamically adapts to these needs by identifying relevant entities and values, then applying suitable prompting strategies such as F-CoT or PoT. In numerically intensive domains, PoT facilitates executable code generation for precise computation.

**Question Complexity.** Temporal QA questions range from direct lookups (e.g., "What year did the team win?") to complex reasoning (e.g., "What was the profit two quarters after policy X?"). These often require temporal anchoring, arithmetic, and sequential logic. SEAR addresses this by decomposing questions and tailoring its strategy based on both table and query characteristics.

**Limitations of Prior Work.** Despite recent interest, most prior work underrepresents the structural and domain diversity seen in real-world tables. Datasets like TempTabQA (Gupta et al., 2023) focus narrowly on specific formats, limiting generalizability. Annotation inconsistencies (Deng et al., 2024) further complicate benchmarking. Symbolic approaches (e.g., DATER (Ye et al., 2023), BINDER (Cheng et al., 2023)) offer logical precision on well-structured tables but falter on hybrid or semi-structured formats. Conversely, text-focused models (e.g., C.L.E.A.R. (Deng et al., 2024)) provide strong language understanding but lack robust symbolic reasoning. These limitations highlight the need for hybrid systems like SEAR, which dynamically integrate symbolic and neural strategies based on task demands.

## 3 Adaptive Reasoning Framework

Humans naturally begin by understanding the objective and analyzing table structures, including cell relationships, headers, and implicit dependencies, while incorporating additional context if available. In temporal tables, this involves identifying both implicit and explicit time-based patterns. Once the problem and context are clear, relevant information is retrieved directly or by decomposing the task into subproblems based on complexity. Finally, logical and numerical reasoning is applied systematically to arrive at a well-founded conclusion.

Motivated by this intuitive approach, we propose the SEAR (Select-Elaborate-Answer & Reasoning) a framework designed to dynamically adapt reasoning strategies based on the structure and complexity of the given table. SEAR builds upon existing prompting methods by introducing a structured, multi-step reasoning process that mirrors human problem solving. It follows a structured three step process to improve temporal table reasoning, ensuring systematic problem solving while leveraging In-context learning for adaptability.

**Step1: Select Crucial Steps** : Identify key reasoning steps without answering directly, creating an efficient problem solving path. Figure 3 shows the actual prompt.

- Problem Understanding: Define the question's objective and analyze table structure.
- Reasoning Process: Select single or multiple strategies from Extract relevant evidence, decompose complex queries, apply logical steps, and generate Python code if needed (when the question involves numerical or arithmetic reasoning. This is guided by the prompt as seen in Figure 3)
- Optimization tips: Simplify steps, retrieve direct answers when possible, and use code for numerical operations.

**Step 2: Elaborate Crucial Steps** : Refine and comprehend selected steps for clarity and effectiveness. Figure 4 shows the actual prompt.

- Add contextual details, specify exact table elements, and refine decomposition.
- Ensure a structured and logically coherent flow toward the final answer.

**Step 3: Answer & Reasoning** : Execute the structured steps to derive an accurate, well-supported answers. Figure 5 shows the actual prompt.

- Follow elaborated steps precisely, referencing extracted evidence.
- Justify answers with logical explanations, when possible directly answer from evidence and integrate Python code for calculations when needed.

| Dataset | Structure | | | Domain | | Reasoning | | Question Types | | | Answer Types | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Flat | Hierarichal | Hybrid | Wikipedia | Finance. | Numerical | Textual | Lookup | Multi-step | Temporal | Long-form | SQL |
| FeTaQA | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ |
| FinQA | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ |
| HiTab[†] | ✗ | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ | ✗ |
| HybridQA | ✗ | ✗ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ |
| MultiHierTT | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ |
| Squall | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ | ✗ | ✗ | ✓ |
| TAT-QA | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ |
| WikiTableQuestions | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ |

Table 1: Comparison of Temporal Table QA datasets by structure, domain, reasoning, and question types. [†]HiTab spans Wikipedia and financial domains. Binary indicators simplify complex question types (e.g., SQL, long-form).

By progressively refining reasoning, SEAR ensures adaptability and robustness across diverse table formats and complexities.

Standard SEAR follows a three-step reasoning process that, while interpretable, introduces additional overhead may impact efficiency. SEAR_Unified simplifies this pipeline by integrating all three steps into a single-pass formulation. Interestingly, this unified approach not only reduces latency and token cost but also achieves higher accuracy, indicating that explicit phase separation may not be necessary for models with strong internal reasoning capacity. SEAR_Unified dynamically selects and refines reasoning steps based on the query and table structure, retrieving key information, decomposing complex queries when needed, and selectively using Python for numerical operations. SEAR_Unified validates intermediate steps and performs error checks to ensure accuracy while reducing redundant complexity. Figures 6 and 7 illustrate the prompt and reasoning path.

We additionally introduce table and context refactoring as a preprocessing step independent of prompting strategy: it converts heterogeneous table formats into a more uniform representation (e.g., Markdown tabular, clarifies headers, aligns units) so that whichever prompting strategy is used, the model has cleaner input. Refactoring is optional but is recommended when the raw table structure is highly variable or noisy; if tables are already well-aligned and simple, one may skip refactoring to save compute/time. Table 2 summarizes the refactoring changes for each dataset and Figure 10 showcase the prompt used.

## 4 Experimental Setup

**Datasets.** We selected eight diverse tabular as shown in table 3 datasets spanning structured, semi-structured, hierarchical, and hybrid tables to ensure a comprehensive evaluation. These datasets present challenges such as entity relations, numerical reasoning, and textual integration, making them well-suited for assessing table reasoning in LLMs as shown in Table 1. For detailed overview of the dataset refer appendix D.

| Categories | fetaqa | finqa | hitab | hybridqa | multi | squall | tatqa | wiki |
|---|---|---|---|---|---|---|---|---|
| **Table Structure** | 1580 | 961 | 616 | 1528 | 1587 | 774 | 2240 | 1503 |
| **Title Clarity** | 1582 | 962 | 386 | 1528 | 1587 | 774 | 2244 | 1504 |
| **Column/Row Header** | 1268 | 919 | 353 | 1229 | 1587 | 774 | 2158 | 1283 |
| **Data Formatting** | 1329 | 957 | 269 | 1476 | 1585 | 774 | 2124 | 1399 |
| **Bolding & Emphasis** | 1207 | 934 | 206 | 1460 | 1524 | 347 | 2200 | 478 |
| **Other** | 328 | 273 | 82 | 468 | 539 | 197 | 696 | 309 |

Table 2: Dataset evaluation for refactoring categories.

*Dataset Filtering:* Adapting TempTabQA's (Gupta et al., 2023) keyword filter (§3.2), we selected temporal cues (e.g., before, year, latest) along with domain-specific terms (e.g., fiscal, quarterly) and applied them across all datasets. This approach reliably captures most of the explicit temporal questions, though purely implicit cases may be missed. Incorporating human judgment could improve coverage but at the cost of scalability.

| Dataset | Brief description | #Qs |
|---|---|---|
| FeTaQA | Wikipedia tables; long-form answers from discontinuous facts | 1,582 |
| FinQA | Financial reports; multi-step numerical reasoning | 962 |
| HiTab | Hierarchical tables; fine-grained numeric questions | 897 |
| HybridQA | Wiki tables + linked text; hybrid reasoning | 1,528 |
| MultiHierTT | Finance; multiple hierarchical tables + long text | 1,587 |
| Squall | WikiTableQ + SQL alignments; structured query tasks | 774 |
| TAT-QA | Finance; tables + text with arithmetic / counting | 2,244 |
| WikiTableQ | Wikipedia trivia; factual + numeric Q over large tables | 1,504 |

Table 3: Number of retained temporal Questions.

**Models:** We used 3 LLM models: GPT4o-mini, Gemini 1.5 Flash, and LLaMA 3.1 70B.

**Prompts & Frameworks:** Effective prompting improves task comprehension and response quality by providing structured instructions. We evaluated 13 prompting strategies spanning direct, structured, temporal, and agentic approaches, as summarized in Table 4.

| Baseline | Brief description | Category |
|---|---|---|
| Chain-of-Thought (COT) (Wei et al., 2022) | Step-by-step natural-language rationale | Direct |
| Evidence Extraction(EE) | Extracts supporting cells first, then answers | Direct |
| Decomposed Prompting(Decomp)(Khot et al., 2023) | Splits complex queries into simpler sub-prompts | Direct |
| Faithful COT (F-COT) (Lyu et al., 2023) | Adds consistency checks to Chain-of-Thought | Direct |
| Program-of-Thought (POT)(Chen et al., 2023) | Generates executable code (e.g., Python) for reasoning | Direct |
| Self-Discover (Zhou et al., 2024) | Model autonomously picks reasoning modules | Structured |
| Self-Ask (Press et al., 2023) | Iteratively asks and answers sub-questions | Structured |
| Plan & Solve (Wang et al., 2023a) | Separates plan generation from execution | Structured |
| C.L.E.A.R. (Deng et al., 2025) | Injects temporal cues for semi-structured tables | Temporal |
| Narration of Thought (NoT) (Zhang et al., 2024) | Requires chronological narration to keep temporal order | Tempooal |
| Self-Consistency Prompting (SCP) (Wang et al., 2023b) | Samples multiple COTs and votes | Agentic |
| Tree of Thought (ToT) (Yao et al., 2023) | Searches a tree of reasoning states with pruning | Agentic |
| Graph of Thought (GoT) (Besta et al., 2023) | Generalises ToT to graph search | Agentic |

Table 4: Prompting baselines grouped by category.

To ensure a balanced evaluation, we included both textual and symbolic reasoning prompts. CoT, Evidence Extraction, and Decomposed Prompting guide models through step-by-step interpretation. SCP augments multiple chains of thought and selects the majority vote. PoT and F-CoT generate structured logic for consistent reasoning. The temporal baselines NoT and C.L.E.A.R. inject explicit chronological cues to help the model track event ordering. The structured baselines Self-Discover, Self-Ask, and Plan & Solve introduce autonomous decomposition and planning to improve reasoning quality. The agentic methods ToT and GoT explore tree- or graph-structured reasoning paths to identify high-value solutions. All methods were evaluated in a few-shot setting except Self-Discover.

**Evaluation:** Evaluating diverse datasets is challenging due to varying answer types, from numerical values to long-form text. A rigid metric may miss semantic correctness, so we propose the *Hybrid Correctness Score (HCS)*, which balances lexical and semantic accuracy by combining Relaxed Exact Match Score (REMS, F1-based) and Contextual Answer Evaluation (CAE, LLM-based). A response is considered correct if its REMS score exceeds 80 or if CAE deems it correct. By integrating both lexical and contextual evaluation, HCS offers a more robust measure of answer correctness. **all reported scores represent HCS** for consistency. Detailed REMS and CAE results are provided in Tables 16 17, 18 in Appendix B.

## 5 Result and Analysis

In this section, we analyze results using Tables (6, 7, 8) which showcase HCS scores.

**Is there a single existing reasoning strategy which works best on all table types?** Performance varies depending on table structure, domain, and question complexity. As observed in Gemini 1.5 Flash results (Table 6), COT performs best on HybridQA, Evidence Extraction excels in HiTab, TATQA, FeTaQA and Squall, while Decomposition is most effective for WikiTabQA and FinQA. POT shows the highest performance in MultiHierTT, whereas F-COT does not emerge as the best baseline in any dataset. A similar trend is evident across GPT and LLaMA models as shown in Table 5. Thus, no single prompting method universally outperforms others, as effectiveness is higly dependent on the dataset's structure and complexity.

| | Gemini 1.5 Flash | GPT 4o mini | Llama 3.1 70B |
|---|---|---|---|
| COT | HybridQA | MultiHierTT, TATQA, FeTaQA | HiTab, HybridQA |
| EE | HiTab, TATQA, FeTaQA | WikiTabQA, HiTab, HybridQA | FeTaQA, Squall |
| Decomp | Squall, WikiTabQA, FinQA | FinQA | WikiTabQA, MultiHierTT, TATQA |
| POT | MultiHierTT | Squall | FinQA |
| F-COT | - | - | - |

Table 5: Dataset for which Baseline reasoning strategy performed best for each model

**Does the Adaptive Reasoning Framework Help?** Table 5 confirms that COT, Evidence Extraction, and Decomposition dominate in most datasets, with POT and F-COT experience improvement in performance for financial and Squall datasets. SEAR dynamically selects its reasoning path, primarily leveraging Evidence Extraction, Decomposition, and Logical Steps (COT) while integrating Python Program for numerical reasoning. by design, it optimally combines dominant reasoning strategies with computation support. SEAR outperforms baseline in 4 dataset for Gemini, in 4 different dataset for GPT, and in 4 different datasets for LLaMA. While SEAR consistently improves performance over baseline across multiple models, it does not generalize equally across all datasets.

**Does unification of SEAR help?** SEAR_Unified optimizes reasoning by merging and refining steps into a single adaptive prompt, reducing overhead while enhancing flexibility. As seen in Table 6, 7 , 8, SEAR_Unified outperforms baselines across all datasets for Gemini, while for GPT and LLaMA, it surpasses baselines in 6 datasets, demonstrating its superiority. This highlights SEAR_Unified's ability to generalize effectively across diverse datasets and models.

We compared our methods with recent structured and modular reasoning approaches, including Self-Discover, Self-Ask, and Plan & Solve. Our approach consistently outperforms these baselines, with particularly strong gains on Multi-HierTT, HiTabs, Squall, and HybridQA. Among them, Self-Discover performs the closest, underscoring the value of modular and adaptive reasoning. We also benchmarked against temporal (NoT, C.L.E.A.R.) and agentic (ToT, GoT, SCP) strategies. Although NoT, C.L.E.A.R., and GoT perform well on FetaQA, TAT-QA, and HiTabs, they fail to deliver consistent improvements on more complex benchmarks.

To evaluate the robustness of our approach we also present aggregated analysis in Table 9 which shows mean Accuracy and mean Regret variance taken across dataset for each method and model. SEAR_U demonstrates consistent performance, achieving highest mean accuracy with notably lowest mean regret variance.

| | wiki | multi | hitab | finqa | tatqa | fetaqa | squall | hybridqa |
|---|---|---|---|---|---|---|---|---|
| COT | 73.60 | 58.79 | 79.04 | 60.08 | 87.30 | 71.30 | 69.90 | 80.76 |
| F-COT | 66.89 | 60.68 | 52.06 | 62.16 | 78.79 | 56.13 | 61.11 | 17.93 |
| Decomp | 78.52 | 61.00 | 75.47 | 62.58 | 91.67 | 67.07 | 67.57 | 74.67 |
| EE | 76.33 | 60.43 | 80.82 | 55.93 | 92.20 | 77.62 | 72.32 | 80.10 |
| PoT | 74.40 | 61.12 | 70.68 | 60.52 | 79.68 | 50.88 | 63.57 | 38.48 |
| NoT | 75.19 | 46.12 | 81.60 | 51.03 | 86.54 | **87.89** | 69.12 | 79.84 |
| ToT | 81.98 | 58.72 | 77.81 | 51.24 | 91.04 | 79.26 | 75.32 | 82.52 |
| GoT | 74.86 | 56.08 | **84.05** | 50.83 | 90.95 | 84.57 | 66.14 | 81.02 |
| SCP | 81.71 | 60.42 | 80.93 | 52.70 | 91.22 | 84.32 | 72.35 | 84.29 |
| CLEAR | 82.71 | 55.57 | 79.71 | 53.95 | **93.27** | 84.00 | 78.81 | **84.48** |
| Self Ask | 78.52 | 45.43 | 79.15 | 64.66 | 81.42 | 80.15 | 70.67 | 63.48 |
| Plan & Solve | 81.72 | 39.51 | 67.56 | 66.32 | 90.60 | 81.83 | 77.00 | 62.63 |
| Self Discover | 80.32 | 59.42 | 78.93 | 65.49 | 91.35 | 81.16 | 74.81 | 80.43 |
| SEAR | 81.45 | 60.18 | 79.71 | 65.90 | 90.02 | 82.87 | 80.23 | 81.15 |
| SEAR_U | 82.18 | **61.75** | 82.61 | **68.71** | 92.78 | 79.84 | **81.52** | 82.00 |
| SEAR+R | 82.71 | 58.54 | 81.05 | 65.49 | 89.39 | 84.20 | 78.04 | 65.90 |
| SEAR_U+R | **83.38** | 56.58 | 82.83 | 67.36 | 91.53 | 85.52 | 77.91 | 67.08 |

Table 6: HCS scores (in %) using Gemini 1.5 Flash, R stands for "Refactoring" and U stands for "Unified".Bold represents the best performer and the underlined represents the second best performer.

| | wiki | multi | hitab | finqa | tatqa | fetaqa | squall | hybridqa |
|---|---|---|---|---|---|---|---|---|
| COT | 78.92 | 57.97 | 77.59 | 64.14 | 92.91 | 84.13 | 67.57 | 78.21 |
| F-COT | 71.61 | 55.32 | 71.35 | 64.97 | 91.04 | 77.81 | 56.46 | 34.62 |
| Decomp | 79.79 | 57.03 | 76.14 | 65.18 | 92.65 | 78.45 | 62.40 | 77.68 |
| EE | 80.12 | 56.77 | 79.38 | 56.03 | 92.81 | 83.88 | 66.67 | 79.58 |
| POT | 79.59 | 57.91 | 76.25 | 56.13 | 90.15 | 72.00 | 72.35 | 61.98 |
| NoT | 65.82 | 44.54 | **80.82** | 50.41 | 88.01 | **85.46** | 52.58 | 76.83 |
| ToT | 81.91 | 56.89 | 79.04 | 55.40 | **96.60** | 82.30 | 66.67 | 80.49 |
| GoT | 71.54 | 52.04 | 74.58 | 51.35 | 90.90 | 81.68 | 53.61 | 75.58 |
| SCP | 79.05 | 57.59 | 79.71 | 55.19 | 92.29 | 84.19 | 66.53 | 80.01 |
| CLEAR | 82.84 | 58.09 | 78.26 | 55.92 | 85.22 | 84.00 | 68.08 | 82.26 |
| Self Ask | 78.66 | 54.38 | 79.60 | 66.11 | 90.76 | 83.03 | 72.09 | 63.48 |
| Plan & Solve | 82.65 | 56.77 | 78.26 | 64.97 | 90.34 | 83.92 | 77.26 | 62.63 |
| Self Discover | 82.71 | 56.46 | 79.60 | 65.70 | 91.67 | 84.51 | 70.28 | **80.43** |
| SEAR | 80.19 | 57.40 | 77.37 | 67.26 | 92.42 | 83.38 | 69.64 | 75.33 |
| SEAR_U | 79.92 | **61.00** | 78.93 | **71.10** | 92.91 | 84.89 | 76.74 | 78.27 |
| SEAR + R | 82.91 | 56.65 | 78.82 | 66.94 | 91.84 | 84.77 | **79.33** | 68.72 |
| SEAR_U + R | **84.18** | 59.29 | 80.27 | 69.75 | 91.44 | 84.39 | 79.20 | 70.48 |

Table 7: HCS scores (in %) using GPT 4o mini, R stands for "Refactoring" and U stands for "Unified".Bold represents the best performer and the underlined represents the second best performer.

**Is table refactoring lossless?** While LLM-based refactoring may introduce a risk of hallucination, we empirically evaluate this using the AutoQA metric (Jain et al., 2024), which measures answer accuracy on both original and refactored tables. As shown in Table 10, the loss in fidelity is minimal. The slight drop in accuracy is primarily due to purposeful modifications, such as the addition of numerical units, adjustments to headers, and revised table titles. Although these changes alter the structure, they improve semantic clarity and enhance the tables' utility for downstream reasoning tasks.

| | wiki | multi | hitab | finqa | tatqa | fetaqa | squall | hybridqa |
|---|---|---|---|---|---|---|---|---|
| COT | 81.05 | 57.59 | 82.95 | 66.22 | 91.00 | 86.03 | 75.45 | 81.66 |
| F-COT | 66.22 | 39.82 | 64.55 | 51.77 | 45.12 | 52.78 | 61.11 | 33.31 |
| Decomp | 82.85 | 59.29 | 81.84 | 65.28 | 93.18 | 84.51 | 73.51 | 80.53 |
| EE | 81.91 | 58.92 | 82.84 | 61.75 | 92.54 | 86.62 | 80.10 | 81.07 |
| POT | 76.53 | 58.98 | 67.56 | 66.42 | 91.40 | 50.44 | 68.22 | 37.76 |
| NoT | 55.57 | 39.76 | 49.83 | 42.23 | 48.57 | 61.18 | 44.85 | 65.32 |
| ToT | 84.57 | 45.35 | 74.99 | 57.58 | 82.67 | 83.50 | 78.29 | 83.18 |
| GoT | 71.27 | 52.61 | 68.45 | 40.24 | 72.73 | **88.49** | 59.19 | 74.80 |
| SCP | 82.96 | 57.80 | 79.38 | 52.52 | 85.22 | 85.46 | 74.96 | 79.75 |
| CLEAR | **86.23** | 54.93 | 76.39 | 56.23 | 92.15 | 86.97 | 79.84 | 79.71 |
| Self Ask | 81.98 | 56.84 | 82.06 | 67.46 | 91.69 | 85.98 | 76.10 | 72.32 |
| Plan & Solve | 82.65 | 55.95 | 80.39 | 66.57 | 92.51 | 83.96 | 76.23 | 70.55 |
| Self Discover | 85.77 | 57.91 | **83.95** | 66.11 | 92.87 | 86.09 | 79.33 | **83.25** |
| SEAR | 82.65 | 59.61 | 83.05 | 66.63 | 92.34 | 85.52 | 81.40 | 79.78 |
| SEAR_U | 82.05 | **62.19** | 82.39 | **70.17** | **93.27** | 87.04 | 82.04 | 80.27 |
| SEAR + R | 82.65 | 57.09 | 82.39 | 67.26 | 91.67 | 86.85 | 76.87 | 67.74 |
| SEAR_U + R | 85.11 | 58.16 | 83.05 | 69.67 | 92.89 | 87.23 | **82.49** | 72.16 |

Table 8: HCS scores (in %) using Llama 3.1 70B, R stands for "Refactoring" and U stands for "Unified".Bold represents the best performer and the underlined represents the second best performer.

**Error Analysis Summary.** We conduct a fine-grained error analysis across six datasets as shown in Figure 2 and find that evidence extraction is the most common failure mode, accounting for the majority of errors in five out of six cases. These errors arise from shallow string matching, ambiguous headers, and missed qualifiers (e.g.,

| | Gemini 1.5 Flash | | GPT-4o mini | | Llama 3.1 70B | |
|---|---|---|---|---|---|---|
| **Method** | Accuracy | Regret | Accuracy | Regret | Accuracy | Regret |
| CoT | 72.60 | 8.03 | 75.18 | 4.91 | 77.74 | 3.51 |
| F-CoT | 56.97 | 23.66 | 65.40 | 14.70 | 51.84 | 29.42 |
| Decomp | 72.32 | 8.31 | 73.66 | 6.43 | 77.62 | 3.63 |
| EE | 74.47 | 6.16 | 74.41 | 5.69 | 78.22 | 3.04 |
| PoT | 62.42 | 18.21 | 70.80 | 9.30 | 64.66 | 16.59 |
| NoT | 72.17 | 8.46 | 68.06 | 12.04 | 50.91 | 30.34 |
| ToT | 74.74 | 5.89 | 74.91 | 5.18 | 73.77 | 7.49 |
| GoT | 73.56 | 7.07 | 68.91 | 11.18 | 65.97 | 15.28 |
| SCP | 75.99 | 4.64 | 74.32 | 5.77 | 74.76 | 6.50 |
| CLEAR | 76.56 | 4.07 | 74.33 | 5.76 | 76.56 | 4.70 |
| Self-Ask | 70.44 | 10.20 | 73.50 | 6.59 | 76.80 | 4.45 |
| Plan & Solve | 70.90 | 9.73 | 74.60 | 5.50 | 76.10 | 5.15 |
| Self-Discover | 76.49 | 4.14 | 76.42 | 3.67 | 79.41 | 1.84 |
| SEAR | 77.69 | 2.94 | 75.37 | 4.72 | 78.87 | 2.38 |
| **SEAR_U** | **78.92** | **1.71** | **77.97** | **2.12** | **79.93** | **1.33** |
| SEAR+R | 75.66 | 4.97 | 76.25 | 3.85 | 76.56 | 4.69 |
| SEAR_U+R | 76.52 | 4.11 | 77.38 | 2.72 | 78.84 | 2.41 |

Table 9: Mean Average and Mean Regret variance across datasets for all 3 models (in %)

| **Dataset** | **fetaqa** | **finqa** | **hitab** | **multi** | **squall** | **tatqa** | **wiki** | **hybridqa** |
|---|---|---|---|---|---|---|---|---|
| **Accuracy** | 99.41 | 95.36 | 98.06 | 88.04 | 86.66 | 99.40 | 96.43 | 84.59 |

Table 10: AutoQA Accuracy after refactoring Tables.

years, units, footnotes), leading models to anchor to plausible but incorrect cells, often before any reasoning or computation can take place. Reasoning errors are more prominent in datasets requiring temporal alignment or multi-hop inference, such as TAT-QA, while code-generation failures dominate in WikiTQ due to parsing issues and faulty aggregation over semi-structured tables. Overall, this suggests that early-stage grounding remains the key bottleneck across tasks, with dataset-specific challenges emerging in reasoning and execution stages. **Please refer to Appendix C** for a detailed breakdown of error types by dataset.

## 6 Discussion

The Adaptive Framework consistently generalizes across multiple datasets by dynamically selecting appropriate reasoning paths. Table 11 summarizes the reasoning paths chosen by GPT-4o-mini, showing that Evidence Extraction is always included. This step helps the model focus on relevant information, aligning with human intuition (Section 3). For lookup-based questions, Evidence Extraction alone suffices, while more complex tasks require a combination of reasoning methods.

Datasets with long-form answers, such as FeTaQA, textual strategies works best. As shown in Table 8, for LLaMA 3.1 70B, FeTaQA achieves higher accuracy with CoT (84.13%) and Decomposed Prompting (78.45%). This trend is further supported by Table 11, where Evidence Extraction

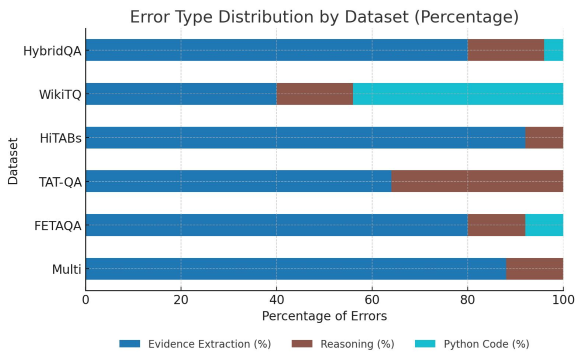


Figure 2: Distribution of error types (evidence extraction, reasoning, and Python code execution) across six benchmark datasets. Evidence extraction emerged as the dominant failure mode in five of the six cases.

+ Decomposed Prompting is the most frequently chosen. Table 12 reinforces this, showing that 87% of FeTaQA's reasoning paths rely on textual methods, highlighting their effectiveness for free-form responses.

FinQA, which is heavy on numerical computation, favors symbolic methods. As seen in Table 8, PoT achieves the best performance, with F-CoT also performing well. Table 11 further confirms this, with Evidence Extraction + F-CoT as the most common reasoning path. Similarly, Table 12 shows that 88.25% of FinQA's reasoning paths involve PoT and F-CoT, emphasizing the strength of symbolic reasoning for computation-heavy datasets.

This pattern extends across datasets, with chosen reasoning paths aligning with their respective strengths. Table 14 and 15 in Appendix B provide reasoning path analysis for LLaMA 3.1 70B and Gemini-1.5-flash, respectively. By dynamically selecting the most effective reasoning approach based on question type and tabular context, the Adaptive Framework consistently delivers strong performance across diverse table structures and reasoning tasks.

**Impact of Table Refactoring.** Refactoring tabular data enhances LLM accuracy by improving clarity, structure, and accessibility. Table 2 categorizes key refactoring techniques that aid model interpretation. In *'Table Structure'*, standardizing tables to Markdown format significantly improves performance. For instance, the Squall dataset, originally in JSON, benefits from this transformation. As shown in Table 7, GPT-4o-mini with SEAR + Refactoring (79.33%) outperforms SEAR (69.64%) by 9.69%, and SEAR_U + Refactoring (79.20%) exceeds SEAR_U (76.74%) by 2.46%. Similarly, LLaMA 3.1 70B achieves its highest accuracy (82.49%) with SEAR_U + Refactoring. In *'Title Clarity'*, refining ambiguous or missing table titles improves context.

Figure 10 illustrates how adding a player's name in the title enhances model comprehension. *'Column/Row Headers'* are refined to eliminate ambiguity and better align entities. *'Data Formatting'* reduces redundant details, such as excessive decimal places, which can increase hallucinations as context size grows (Liu et al., 2023). Limiting decimals helps models focus and improves accuracy. *'Bolding and Emphasis'* highlights key details, directing the model's attention to relevant content. *'Other'* refinements, such as adding units, removing whitespace, and reformatting text, further enhance readability. The prompt for table refactoring is shown in Figure 9.

Refactoring yields the most consistent improvements for clean or moderately structured datasets such as WikiTableQA, HiTab, FinQA, FeTaQA, and Squall, where formatting and clarity directly enhance interpretability. In contrast, highly complex or hybrid datasets (e.g., HybridQA, MultiHierTT) showed negative effects, indicating that refactoring is best treated as an optional, data-dependent step rather than a universal enhancement.

Finally, refactoring introduces a computational cost, adding one LLM call per table. While lightweight, this step should be applied judiciously based on data complexity and expected gains.

| **Reasoning Path** | **Datasets** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **fetaqa** | **finqa** | **hitab** | **hybridqa** | **multi** | **squall** | **tatqa** | **wiki** |
| **EE** | 175 | 46 | 476 | **1332** | 194 | 13 | **929** | **703** |
| **EE,Decomp** | **1365** | 65 | 191 | 28 | 127 | 160 | 249 | 293 |
| **EE,F-COT** | 23 | **703** | 111 | 5 | **335** | **581** | **547** | 246 |
| **EE,POT** | 9 | 138 | 107 | 143 | **909** | 14 | **482** | 186 |
| **COT,EE** | 1 | 1 | 4 | 12 | 5 | - | 5 | 32 |
| **COT,EE,Decomp** | 8 | 1 | 3 | 2 | - | 1 | 1 | 13 |
| **COT,EE,F-COT** | 1 | 7 | 1 | - | 5 | 5 | 12 | 17 |
| **COT,EE,POT** | - | 1 | 4 | 6 | 12 | - | 19 | 14 |
| **Total** | 1582 | 962 | 897 | 1528 | 1587 | 774 | 2244 | 1504 |

Table 11: Reasoning Path distribution for GPT-4o-mini.

| **Dataset** | **COT** | | **EE** | | **Decomp** | | **POT** | | **F-COT** | |
|---|---|---|---|---|---|---|---|---|---|---|
| | # | % | # | % | # | % | # | % | # | % |
| **fetaqa** | 10 | 0.63 | 1582 | 100 | **1373** | **86.79** | 9 | 0.57 | 24 | 1.52 |
| **finqa** | 10 | 1.03 | 962 | 100 | 66 | 6.86 | 139 | 14.45 | **710** | **73.8** |
| **hitab** | 12 | 1.34 | 897 | 100 | **194** | **21.63** | 111 | 12.38 | 112 | 12.49 |
| **hybridqa** | 20 | 1.31 | 1528 | 100 | 30 | 1.96 | **149** | **9.75** | 5 | 0.33 |
| **multi** | 22 | 1.39 | 1587 | 100 | 127 | 8.01 | **921** | **58.03** | 132 | 8.32 |
| **squall** | 6 | 0.78 | 774 | 100 | 161 | 20.8 | 14 | 1.81 | **586** | **75.71** |
| **tatqa** | 37 | 1.65 | 2244 | 100 | 250 | 11.14 | 501 | 22.33 | **559** | **24.91** |
| **wiki** | 76 | 5.05 | 1504 | 100 | **306** | **20.35** | 200 | 13.3 | 263 | 17.49 |

Table 12: Distribution of reasoning methods across all the datasets for GPT-4o-mini.

**SEAR in the Context of Agentic Frameworks.** Agentic frameworks have gained attention for their ability to handle complex reasoning tasks through modular, interacting components such as planning, memory retrieval, and tool use. Although SEAR is not agentic by design, its structured reasoning process aligns with the modular philosophy of agentic systems. Each SEAR module could be instantiated as an individual agent within such a framework. However, the goal of this work is to explore how far prompting alone without external tools or orchestration can be used to address temporal table QA. This design choice prioritizes simplicity and self-containment. Importantly, the central challenge SEAR addresses is selecting and sequencing the appropriate reasoning strategies for a given question and table structure remains critical even within agentic systems.

While agentic architectures offer a more general execution framework, they still depend on effective strategy selection. In this sense, SEAR provides a complementary perspective, offering insights into reasoning decomposition that could inform or enhance agent-based designs.

## 7 Related Work

**Tabular Reasoning.** LLMs have been widely applied to tabular reasoning tasks such as question answering, semantic parsing, and table-to-text generation (Chen et al., 2020a; Gupta et al., 2020; Zhang et al., 2020; Zhang and Balog, 2020). Early approaches like TAPAS (Herzig et al., 2020), TaBERT (Yin et al., 2020), and TABBIE (Iida et al., 2021) improve table comprehension by integrating tabular and textual embeddings, allowing models to better process structured information. Other methods, such as Table2Vec (Zhang et al., 2019) and TabGCN (Pramanick and Bhattacharya, 2021), explore alternative tabular representations, enhancing LLMs' ability to infer relationships between table elements. However, these methods primarily focus on structured tables and do not explicitly address temporal reasoning, which introduces additional complexity when reasoning over tabular data.

**Symbolic Reasoning for Tables.** Recent work has explored symbolic reasoning for structured tables with predefined schemas, improving logical inference and data consistency (Cheng et al., 2023; Ye et al., 2023; Wang et al., 2024). These methods rely on well-defined structures to extract and process information effectively. However, they struggle with semi-structured and hierarchical tables, where relationships between data points are implicit rather than explicitly defined.

**Other Reasoning Frameworks.** C.L.E.A.R (Deng et al., 2024) demonstrated strong temporal reasoning on domain-specific semi-structured tables by integrating domain knowledge into responses. Similarly, Meta-Reasoning Prompting (MRP)(Gao et al., 2024) selects the optimal reasoning strategy through a two-step process but does not combine reasoning techniques for complex tasks. In contrast, our approach integrates both textual and symbolic reasoning to enhance performance across diverse table types while dynamically selecting the best reasoning path. Moreover, our SEAR-Unified prompt streamlines this into a single-step process, ensuring efficiency and consistency across different table structures.

**Comparative Analysis.** To situate SEAR and SEAR_Unified within the broader landscape of prompting methods, we provide a systematic comparison in Table 13. Single-inference methods such as CoT and PoT offer simplicity, but lack structured reasoning phases for complex table QA. Tree-based methods (ToT, GoT) explore multiple paths, but incur high computational costs. Multi-step approaches like Self-Ask enable iterative refinement, while Self-Discover selects reasoning strategies, yet neither provides explicit structure for temporal and numerical reasoning. In contrast, SEAR uniquely combines modularity, selective code generation, and explicit three-phase reasoning, allowing systematic optimization of individual components. SEAR_Unified maintains this reasoning structure implicitly in a single inference step, achieving both efficiency and effectiveness.

| **Method** | **Inference Steps** | **Modular** | **Code Generation** |
|---|---|---|---|
| CoT | Single step | No | No |
| EE | Single step | No | No |
| Decomp | Single-step | No | No |
| F-CoT | Single-step | No | Yes |
| PoT | Single-step | No | Yes |
| NoT | Single-step | No | No |
| ToT | Single-step | No | No |
| GoT | Single-step | No | No |
| SCP | Single-step | No | No |
| CLEAR | Single-step | No | No |
| Self-Ask | Multi-step | No | No |
| Plan & Solve | Single-step | No | No |
| Self-Discover | Multi-step | Yes | No |
| **SEAR** | **Multi-step** | **Yes** | **Yes** (Selective) |
| **SEAR_U** | **Single-step** | **Yes** | **Yes** (Selective) |

Table 13: Related work matrix comparing SEAR and SEAR_Unified with major prompting paradigms.

## 8 Conclusion and Future Work

This paper introduces SEAR, an adaptive reasoning strategy for LLMs to tackle TTQA tasks, along with its consolidated version, SEAR_Unified. Additionally, we take a step toward a unified table representation by incorporating table refactoring as an enhancement. Our study provides a comprehensive analysis of various reasoning strategies across eight diverse datasets, benchmarking SEAR and SEAR_Unified against multiple baselines.

Results demonstrate that SEAR, SEAR_Unified and with Table Refactoring significantly outperforms popular LLM reasoning methods, with SEAR_Unified surpassing SEAR itself, showcasing its ability to optimize and streamline reasoning with minimal overhead. This highlights capability of modern LLMs to dynamically adjust reasoning within a single prompt, reducing the need for explicit multi-step processes. Our findings reinforce the importance of adaptive reasoning and structured table representation, paving the way for further advancements in LLM-based temporal table reasoning.

While SEAR-based approaches have significantly improved Temporal Table QA, several areas remain open for further exploration. In this work, we have explored Markdown as a unified tabular representation, exploring alternative formats such as JSON, CSV, or HTML may further improve adaptability across diverse table structures. Currently, our experiments relied on in-context learning, which can limit scalability and efficiency. Future work should explore lightweight adaptive reasoning techniques with self-refinement loops, building on the flexibility demonstrated by SEAR. Lastly, valuating SEAR-based methods on additional domains, such as medical or scientific evolution datasets, would help validate the robustness of adaptive reasoning strategies for LLMs.

## Limitations

While our study has yielded interesting observations, it's crucial to acknowledge its limitations. A closer look at the HCS scores in Table 6, 7, 8, reveals that while improvements are observed for datasets with single table contexts, datasets containing multiple tables, such as MultiHierTT and Hybrid tables, show a decline in performance with SEAR-based approaches. This highlights a key limitation of our Table Refactoring method, suggesting that restructuring strategies may need further refinement to handle multi-table contexts effectively. Additionally, scalability remains a concern, as our approach relies on In-Context Learning (ICL), which may not scale effectively for large table datasets. The reliance on ICL-based reasoning can lead to performance bottlenecks.

## Ethics Statement

We confirm that our work adheres to the highest ethical standards in research and publication. We will publicly release our code and filtered datasets to enable the research community to validate and build upon our findings. We are committed to the responsible and fair use of computational linguistics methodologies. The claims in our paper accurately reflect the experimental results. While using black-box large language models introduces some stochasticity, we mitigate this by maintaining a fixed temperature. We utilize an AI assistive tools for writing while ensuring absence of bias. We provide comprehensive details on annotations, dataset splits, models used, and prompting methods tried, ensuring the reproducibility of our work.

## Acknowledgement

This research has been supported in part by the ONR Contract N00014-23-1-2364, and conducted as a collaborative effort between *Arizona State University* and the *University of Pennsylvania*. We gratefully acknowledge the Complex Data Analysis and Reasoning Lab at School of Augmented Intelligence, *Arizona State University* for providing computational resources and institutional support. We also thank the anonymous reviewers for their thoughtful feedback and constructive suggestions. We extend special appreciation to our lab cat, Coco, whose presence helped both our team and our professor maintain just the right balance of focus and levity during deadlines.

## References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2023. Graph of thoughts: Solving elaborate problems with large language models. *Preprint*, arXiv:2308.09687.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Preprint*, arXiv:2211.12588.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020a. Tabfact: A large-scale dataset for table-based fact verification. *Preprint*, arXiv:1909.02164.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020b. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1026–1036, Online. Association for Computational Linguistics.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2022. 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. *Preprint*, arXiv:2210.02875.

Irwin Deng, Kushagra Dixit, Vivek Gupta, and Dan Roth. 2024. Enhancing temporal understanding in llms for semi-structured tables. *Preprint*, arXiv:2407.16030.

Irwin Deng, Kushagra Dixit, Dan Roth, and Vivek Gupta. 2025. Enhancing temporal understanding in LLMs for semi-structured tables. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4936–4955, Albuquerque, New Mexico. Association for Computational Linguistics.

Peizhong Gao, Ao Xie, Shaoguang Mao, Wenshan Wu, Yan Xia, Haipeng Mi, and Furu Wei. 2024. Meta reasoning for large language models. *Preprint*, arXiv:2406.11698.

Vivek Gupta, Pranshu Kandoi, Mahek Vora, Shuo Zhang, Yujie He, Ridho Reinanda, and Vivek Srikumar. 2023. TempTabQA: Temporal question answering for semi-structured tables. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2431–2453, Singapore. Association for Computational Linguistics.

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*. Association for Computational Linguistics.

Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. Tabbie: Pretrained representations of tabular data. *Preprint*, arXiv:2105.02584.

Parag Jain, Andreea Marzoca, and Francesco Piccinno. 2024. STRUCTSUM generation for faster text comprehension. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7876–7896, Bangkok, Thailand. Association for Computational Linguistics.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. *Preprint*, arXiv:2210.02406.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *Preprint*, arXiv:2307.03172.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 305–329, Nusa Dua, Bali. Association for Computational Linguistics.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Nick Schoelkopf, Riley Kong, Xiangru Tang, Murori Mutuma, Ben Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, and Dragomir Radev. 2021. Fetaqa: Free-form table question answering. *Preprint*, arXiv:2104.00369.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *Preprint*, arXiv:1508.00305.

Aniket Pramanick and Indrajit Bhattacharya. 2021. Joint learning of representations for web-tables, entities and types using graph convolutional network. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1197–1206, Online. Association for Computational Linguistics.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

Ansh Radhakrishnan, Karina Nguyen, Anna Chen, Carol Chen, Carson Denison, Danny Hernandez, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Sam McCandlish, Sheer El Showk, Tamera Lanham, Tim Maxwell, Venkatesa Chandrasekaran, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. 2023. Question decomposition improves the faithfulness of model-generated reasoning. *Preprint*, arXiv:2307.11768.

Tianze Shi, Chen Zhao, Jordan Boyd-Graber, Hal Daumé III, and Lillian Lee. 2020. On the potential of lexico-logical alignments for semantic parsing to SQL queries. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1849–1864, Online. Association for Computational Linguistics.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023a. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*.

Yuqing Wang and Yun Zhao. 2024. TRAM: Benchmarking temporal reasoning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6389–6415, Bangkok, Thailand. Association for Computational Linguistics.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2024. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *Preprint*, arXiv:2401.04398.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Siheng Xiong, Ali Payani, Ramana Kompella, and Faramarz Fekri. 2024. Large language models can learn temporal reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10452–10470, Bangkok, Thailand. Association for Computational Linguistics.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decompose evidence and questions for table-based reasoning. *Preprint*, arXiv:2301.13808.

Pengcheng Yin, Graham Neubig, Wen tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. *Preprint*, arXiv:2005.08314.

Li Zhang, Shuo Zhang, and Krisztian Balog. 2019. Table2vec: Neural word and entity embeddings for table population and retrieval. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '19. ACM.

Shuo Zhang and Krisztian Balog. 2020. Web table extraction, retrieval, and augmentation: A survey. *ACM Trans. Intell. Syst. Technol.*, 11(2):13:1–13:35.

Shuo Zhang, Zhuyun Dai, Krisztian Balog, and Jamie Callan. 2020. Summarizing and exploring tabular data in conversational search. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pages 1537–1540, New York, NY, USA. Association for Computing Machinery.

Xinliang Frederick Zhang, Nicholas Beauchamp, and Lu Wang. 2024. Narrative-of-thought: Improving temporal reasoning of large language models via recounted narratives. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, Miami, Florida. Association for Computational Linguistics.

Xuanliang Zhang, Dingzirui Wang, Longxu Dou, Qingfu Zhu, and Wanxiang Che. 2025. A survey of table reasoning with large language models. *Front. Comput. Sci.*, 19(9).

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. 2022. Multihiertt: Numerical reasoning over multi hierarchical tabular and textual data. *Preprint*, arXiv:2206.01347.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. 2024. Self-discover: Large language models self-compose reasoning structures. *Preprint*, arXiv:2402.03620.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. *Preprint*, arXiv:2105.07624.

## A Prompt Examples

This section contains the Prompts example for standart 3-step SEAR (Figure 3, 4, 5), SEAR Unified (Figure 6 7), Evaluation Prompt (Figure 8) and Refactoring Prompt and Response (Figure 9, 10).

## B REMS and CAE Results

This section contains the additional result for Reasoning Path Distribution for Llama and Gemini (Table 14, 15) and also contains complete result for CAE and REMS score for all GPT, Gemini and Llama (Table 16, 17, 18).

*1. Relaxed Exact Match Score(REMS):* This metric uses an F1-score to measure token overlap between the predicted and gold answer, allowing partial matches for better precision-recall balance. Unlike strict exact match, REMS is more flexible with lexical variations. For numerical answers, it permits a ±5% tolerance after decimal instead of token matching. For example, if the correct answer is 10.64, a prediction of 10.62 is accepted, while 11.64 is not.

Despite its flexibility, REMS does not always reflect true semantic accuracy. High scores indicate strong token alignment, but valid paraphrases can be unfairly penalized. For instance, the correct answer "Barack Obama was the 44th President of the United States" would receive a high score for "Obama was the 44th U.S. President" due to token overlap, but "Obama, a politician, led the U.S." may score lower despite being factually correct. This limitation makes careful interpretation

*2. Contextual Answer Evaluation(CAE):* CAE is an LLM-based scoring method that assesses responses based on meaning rather than exact token overlap. Using a carefully crafted prompt, it determines whether a response correctly conveys the intended information. Unlike traditional lexical matching, CAE accounts for paraphrasing and rewording, ensuring a more nuanced assessment of correctness, particularly for complex or free-form answers. The full CAE prompt used for evaluation is provided in Figure 8

| **Reasoning Path** | **fetaqa** | **finqa** | **hitab** | **hybridqa** | **multi** | **squall** | **tatqa** | **wiki** |
|---|---|---|---|---|---|---|---|---|
| **EE** | 221 | 39 | 561 | 1072 | 356 | 9 | 1040 | 987 |
| **EE,Decomp** | 553 | 21 | 19 | 8 | 33 | 28 | 59 | 81 |
| **EE,F-COT** | 571 | 853 | 123 | 35 | 262 | 709 | 391 | 236 |
| **EE,POT** | 234 | 45 | 194 | 405 | 919 | 25 | 753 | 187 |
| **COT,EE** | - | - | - | 6 | 5 | - | 1 | 7 |
| **COT,EE,Decomp** | 3 | - | - | 2 | 10 | 1 | - | 2 |
| **COT,EE,F-COT** | - | 3 | - | - | - | 2 | - | 4 |
| **POT** | - | 1 | - | - | 2 | - | - | - |
| **Total** | 1582 | 962 | 897 | 1528 | 1587 | 774 | 2244 | 1504 |

Table 14: Reasoning Path distribution across all datasets for Llama 3.1 70B.

| **Reasoning Path** | **fetaqa** | **finqa** | **hitab** | **hybridqa** | **multi** | **squall** | **tatqa** | **wiki** |
|---|---|---|---|---|---|---|---|---|
| **EE** | 982 | 106 | 675 | 1492 | 155 | 112 | 1160 | 875 |
| **EE,DecompE** | 197 | 16 | 6 | 2 | 87 | 17 | 9 | 186 |
| **EE,F-COT** | 175 | 796 | 29 | - | 333 | 516 | 49 | 173 |
| **EE,POT** | 191 | 42 | 186 | 33 | 1010 | 119 | 1025 | 268 |
| **COT,EE** | 25 | - | - | 1 | - | 1 | 1 | 2 |
| **COT,EE,Decomp** | 3 | - | - | - | - | 1 | - | - |
| **COT,EE,F-COT** | 2 | 1 | - | - | - | 6 | - | - |
| **COT,EE,POT** | 7 | - | 1 | - | - | 1 | - | - |
| **Decomp** | - | 1 | - | - | - | - | - | - |
| **POT** | - | - | - | - | 2 | 1 | - | - |
| **Total** | 1582 | 962 | 897 | 1528 | 1587 | 774 | 2244 | 1504 |

Table 15: Reasoning Path distribution across all datasets for Gemini-1.5-Flash.

| | wiki | | multi | | hitab | | finqa | | tatqa | | fetaqa | | squall | | hybridqa | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | REMS | CAE | REMS | CAE | REMS | CAE | REMS | CAE | REMS | CAE | REMS | CAE | REMS | CAE | REMS | CAE |
| COT | 77.31 | 76.66 | **57.49** | 49.72 | 75.26 | 74.58 | 58.94 | 57.07 | 81.68 | 87.48 | 28.38 | 84.13 | 66.27 | 65.25 | 74.07 | 76.51 |
| F–COT | 67.85 | 67.82 | 49.39 | 51.35 | 41.44 | 69.79 | 60.78 | 61.12 | 67.36 | 86.76 | **40.46** | 77.69 | 52.97 | 53.36 | 29.78 | 32.79 |
| Decomp | 77.69 | 76.60 | 56.12 | 49.02 | 73.19 | 73.36 | 60.40 | 58.21 | 86.13 | 87.25 | 28.71 | 78.45 | 61.07 | 59.30 | 74.71 | 74.87 |
| EE | 78.57 | 77.86 | 56.32 | 48.27 | **76.16** | 76.92 | 50.94 | 46.88 | **90.22** | 88.06 | 28.42 | 83.82 | 65.55 | 64.60 | **75.85** | **76.96** |
| POT | 76.28 | 75.93 | 53.41 | 53.12 | 41.92 | 73.47 | 51.88 | 52.49 | 66.88 | 86.10 | 29.71 | 72.00 | 65.90 | 69.12 | 58.66 | 60.27 |
| NoT | 63.07 | 63.56 | 39.04 | 38.44 | 69.96 | 76.25 | 44.22 | 46.05 | 72.78 | 82.58 | 29.23 | 85.46 | 51.11 | 50.52 | 72.17 | 75.65 |
| ToT | 79.79 | 78.92 | 53.00 | 52.43 | 68.39 | 76.92 | 51.96 | 49.90 | 81.13 | 88.01 | 30.15 | 82.17 | 64.98 | 63.44 | 76.96 | 78.01 |
| GoT | 69.33 | 67.09 | 48.80 | 45.05 | 65.91 | 71.68 | 47.41 | 46.36 | 83.30 | 86.14 | 28.95 | 81.67 | 52.67 | 48.58 | 71.79 | 72.05 |
| SCP | 77.10 | 76.73 | 56.44 | 52.11 | 74.58 | 77.15 | 51.90 | 50.00 | 84.71 | 86.63 | 28.51 | 84.13 | 64.71 | 64.47 | 75.49 | 78.73 |
| CLEAR | 80.23 | 79.72 | 52.67 | 57.40 | 68.62 | 75.81 | | | 85.23 | 91.13 | 29.28 | 83.94 | 65.85 | 66.28 | 77.98 | 79.84 |
| Self ASK | 75.76 | 75.99 | 48.96 | 51.66 | 67.79 | 77.48 | 57.65 | 63.61 | 80.57 | 87.01 | 29.46 | 83.03 | 65.40 | 69.12 | 66.86 | 68.39 |
| Self Discover | 79.91 | 79.54 | 51.85 | 53.62 | 67.29 | 77.92 | 58.24 | 62.78 | 78.72 | 88.45 | 29.38 | 84.51 | 68.50 | 67.05 | 77.16 | 79.90 |
| Plan and Solve | 79.52 | 79.25 | 52.52 | 53.68 | 69.92 | 76.14 | 57.83 | 61.85 | 80.39 | 86.72 | 28.85 | 83.71 | 74.09 | 74.80 | 66.13 | 66.55 |
| SEAR | 78.32 | 76.60 | 54.70 | 50.98 | 67.36 | 74.58 | 62.52 | 60.91 | 81.94 | 85.83 | 29.53 | 83.38 | 67.56 | 60.72 | 72.07 | 73.63 |
| SEAR_U | 77.50 | 77.53 | 56.39 | **56.84** | 71.78 | 76.70 | **62.87** | **67.57** | 88.31 | **89.75** | 31.06 | **84.89** | 72.26 | 73.77 | 74.96 | 75.85 |
| SEAR + R | 80.51 | 79.39 | 54.04 | 51.10 | 68.40 | 75.92 | 61.88 | 60.08 | 81.63 | 85.87 | 29.71 | 84.39 | **76.85** | 74.03 | 65.89 | 66.03 |
| SEAR_U + R | **81.14** | **81.25** | 55.54 | 55.51 | 72.13 | **77.59** | 62.43 | 66.53 | 86.56 | 88.23 | 30.47 | 84.70 | 76.21 | **76.87** | 66.96 | 67.74 |

Table 16: REMS & CAE score (in %) for all reasoning strategies across all datasets using GPT-4o mini. R stands for "Refactoring," U for "Unified."

| | wiki | | multi | | hitab | | finqa | | tatqa | | fetaqa | | squall | | hybridqa | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | REMS | CAE | REMS | CAE | REMS | CAE | REMS | CAE | REMS | CAE | REMS | CAE | REMS | CAE | REMS | CAE |
| COT | 71.86 | 71.28 | 57.29 | 39.26 | 73.97 | 74.25 | 58.00 | 39.29 | 80.81 | 85.34 | 28.25 | 71.24 | 69.44 | 69.66 | 77.29 | 76.57 |
| F-COT | 64.76 | 57.51 | 58.36 | 47.83 | 35.68 | 49.34 | 60.60 | 34.20 | 64.86 | 74.88 | **37.05** | 55.69 | 60.40 | 60.73 | 17.86 | 15.97 |
| Decomp | 76.26 | 75.00 | 58.90 | 41.84 | 71.70 | 72.44 | 60.72 | 32.22 | 84.23 | 85.12 | 29.91 | 67.07 | 66.01 | 65.98 | 72.94 | 69.31 |
| EE | 74.24 | 72.81 | 59.02 | 42.41 | 74.61 | 76.43 | 54.54 | 30.46 | **86.14** | 86.27 | 28.63 | 77.62 | 71.89 | 72.03 | 74.12 | 68.72 |
| POT | 72.65 | 66.69 | **60.00** | 47.01 | 41.37 | 67.54 | 54.90 | 58.10 | 66.74 | 75.61 | 26.73 | 50.88 | 62.66 | 62.99 | 38.18 | 33.84 |
| NoT | 70.40 | 73.07 | 40.59 | 41.46 | 72.08 | 75.59 | 58.32 | 64.24 | 69.41 | 76.69 | 30.73 | 87.99 | 65.03 | 67.05 | 75.43 | 77.68 |
| ToT | 79.79 | 78.92 | 54.65 | 54.88 | 70.33 | 75.70 | 41.68 | 47.40 | 76.84 | 86.68 | 29.48 | 79.20 | 71.90 | 73.00 | 79.42 | 80.17 |
| GoT | 72.82 | 70.88 | 51.20 | 50.03 | 68.41 | 82.16 | 48.34 | 46.57 | 79.15 | 85.34 | 29.57 | 84.45 | 62.08 | 63.05 | 77.14 | 78.53 |
| SCP | 79.40 | 78.92 | 57.72 | 56.27 | 75.90 | 78.37 | 46.53 | 46.26 | 81.38 | 86.72 | 27.85 | 84.32 | 68.82 | 70.80 | 79.26 | 82.00 |
| CLEAR | 79.82 | 79.79 | 56.08 | 0.06 | 70.38 | 76.92 | 49.47 | 48.13 | 79.94 | 90.42 | 28.63 | 83.94 | 75.32 | 77.26 | 79.59 | 81.81 |
| Self ASK | 76.25 | 75.79 | 40.23 | 42.59 | 69.07 | 75.80 | 57.32 | 62.68 | 66.92 | 78.65 | 28.61 | 83.21 | 67.83 | 68.60 | 59.47 | 61.25 |
| Self Discover | 77.80 | 77.26 | 54.34 | 55.82 | 70.63 | 76.58 | 58.62 | 62.68 | 74.31 | 88.10 | 27.49 | 81.09 | 69.78 | 72.86 | 76.24 | 77.55 |
| Plan and Solve | 79.09 | 78.59 | 34.20 | 37.05 | 59.51 | 64.65 | 58.31 | 64.24 | 75.49 | 87.61 | 26.69 | 80.82 | 74.15 | 74.28 | 58.91 | 59.81 |
| SEAR | 79.08 | 78.19 | 57.15 | 54.69 | 74.93 | 76.81 | 59.90 | 61.02 | 75.07 | 83.87 | 28.75 | 82.87 | 76.14 | 68.60 | **77.61** | 78.08 |
| SEAR_U | 79.32 | 80.32 | 59.27 | **57.34** | 78.53 | 79.38 | **63.16** | **65.59** | 82.70 | **86.68** | 31.57 | 79.77 | **77.29** | **79.59** | 77.11 | **79.84** |
| SEAR + R | 80.27 | 78.46 | 55.32 | 52.30 | 75.08 | 77.37 | 59.88 | 60.50 | 73.57 | 84.54 | 28.97 | 84.20 | 76.13 | 72.09 | 62.43 | 62.24 |
| SEAR_U + R | **80.78** | **81.32** | 53.09 | 53.62 | **78.94** | **79.60** | 61.98 | 63.83 | 82.20 | 85.65 | 32.89 | **85.52** | 75.16 | 75.97 | 62.96 | 64.86 |

Table 17: REMS & CAE score (in %) for all reasoning strategies across all datasets using Gemini1.5 Flash. R stands for "Refactoring," U for "Unified."

## SEAR: Step 1

You are a adaptive reasoner tasked with constructing the most efficient pathway for solving tabular questions. Your goal is to select or create minimal, high-level steps to guide reasoning, avoiding direct answers. NOTE - Do not answer, only select crucial steps.

Guidelines:
Problem Understanding:
Identify Objective: Define the question's goal.
Comprehend Problem: Understand the scope and nature of the problem.

Reasoning Process:
Evidence Extraction: Extract relevant rows, columns, and text.
Decomposition: Break down complex questions into sub-questions if necessary.
Step-by-Step Reasoning: Apply logical steps to solve sub-questions or the main problem.
Python Code Generation: Opt to generate code (single or multiple scripts) if calculations are required.

Optimization Tips:
Direct Answer Path: Use evidence extraction to find the answer directly, when possible.
Simplify: Break down complex questions into simpler components.
Code Integration: Include Python code generation for essential calculations.

Few examples are given below with their respective crucial steps selected from the meta-reasoning process. Each example contains its own table, text, and question. Interpret the problem and select only the most essential steps for reaching to answer.

Table:

```
Model          | 2005   | 2006   | 2007   | 2008   | 2009   | 2010   | 2011   | 2012   | 2013   |
Škoda Octavia  | 233322 | 270274 | 309951 | 344857 | 317335 | 349746 | 387200 | 409360 | 359600 |
Škoda Fabia    | 236698 | 243982 | 232890 | 246561 | 264173 | 229045 | 266800 | 255025 | 202000 |
Škoda Superb   | 22091  | 20989  | 20530  | 25645  | 44548  | 98873  | 116700 | 106847 | 94400  |
Škoda Roomster |        | 14422  | 66661  | 57467  | 47152  | 32332  | 36000  | 39249  | 33300  |
Škoda Yeti     |        |        |        |        | 11018  | 52604  | 70300  | 90952  | 82400  |
Škoda Rapid    |        |        |        |        |        |        | 1700   | 9292   | 103800 |
Škoda Citigo   |        |        |        |        |        |        | 509    | 36687  | 45200  |
```

Question: How many Skoda cars were sold in 2010?

LLM Output

Crucial Steps:

Identify Objective: Define the goal.
Evidence Extraction:Extract relevant rows, columns, and text.
Python Code Generation: Generate single Python code to sum the extracted values.

Figure 3: Sear Step 1 Prompt Example

| | wiki | | multi | | hitab | | finqa | | tatqa | | fetaqa | | squall | | hybridqa | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | REMS | CAE | REMS | CAE | REMS | CAE | REMS | CAE | REMS | CAE | REMS | CAE | REMS | CAE | REMS | CAE |
| COT | 79.20 | 78.86 | 56.91 | 48.71 | 80.77 | **81.38** | 60.91 | 60.81 | 83.69 | 86.10 | 28.07 | 86.03 | 73.21 | 73.39 | 79.10 | **79.78** |
| F-COT | 63.02 | 62.43 | 37.21 | 37.30 | 37.35 | 61.76 | 48.14 | 48.44 | 59.67 | 61.72 | 25.34 | 52.72 | 56.53 | 58.01 | 30.30 | 31.28 |
| Decomp | 80.71 | 80.78 | 58.39 | 52.24 | 78.71 | 80.72 | 60.50 | 59.77 | 86.62 | 86.41 | 29.36 | 84.51 | 71.00 | 71.58 | **79.98** | 77.75 |
| EE | 80.30 | 79.79 | 57.70 | 48.27 | **81.42** | 80.05 | 57.03 | 53.53 | 89.09 | 87.70 | 28.63 | 86.62 | 78.12 | 77.78 | 78.33 | 78.73 |
| POT | 74.74 | 73.34 | 56.47 | 55.14 | 37.05 | 65.44 | 62.44 | 61.75 | 65.02 | 87.17 | 20.25 | 50.44 | 63.43 | 64.73 | 35.63 | 35.60 |
| NoT | 51.86 | 52.39 | 30.77 | 34.85 | 43.25 | 46.82 | 33.88 | 39.50 | 41.53 | 46.75 | 20.86 | 61.19 | 44.86 | 47.03 | 68.12 | 69.50 |
| ToT | 82.28 | 81.72 | 40.89 | 46.06 | 78.48 | 80.71 | 55.79 | 49.06 | 86.05 | 90.01 | 29.13 | 83.44 | 74.88 | 75.97 | 78.24 | 80.96 |
| GoT | 69.34 | 68.02 | 50.49 | 48.08 | 65.25 | 66.33 | 36.59 | 40.24 | 72.59 | 77.58 | 30.22 | 88.50 | 59.04 | 61.88 | 70.42 | 73.23 |
| SCP | 82.57 | 85.10 | 55.19 | 59.48 | 80.15 | 84.05 | 52.65 | 51.98 | 84.19 | 90.06 | 28.68 | 85.40 | 77.03 | 77.39 | 77.15 | 79.71 |
| CLEAR | 83.49 | 82.91 | 83.50 | 85.95 | 83.50 | 85.95 | 36.50 | 42.20 | 90.06 | 92.15 | 29.36 | 86.92 | 77.39 | 79.84 | 75.58 | 77.55 |
| SEAR | 80.69 | 78.79 | 57.76 | 50.79 | 75.45 | 78.60 | 61.40 | 60.40 | 84.67 | **88.41** | 29.47 | 85.52 | 78.74 | 72.22 | 76.43 | 77.29 |
| SEAR_U | 78.91 | 79.26 | **60.02** | **58.03** | 75.12 | 79.38 | **63.30** | **66.01** | **89.20** | 86.36 | 34.15 | 87.04 | 78.74 | 80.62 | 77.11 | 78.24 |
| SEAR + R | 80.17 | 78.46 | 54.97 | 48.02 | 75.77 | 78.37 | 62.00 | 61.43 | 81.71 | 86.99 | 29.53 | 86.85 | 73.95 | 70.67 | 67.35 | 70.75 |
| SEAR_U + R | **82.53** | **82.05** | 56.15 | 52.68 | 76.19 | 77.70 | 61.66 | 66.03 | 86.58 | 86.47 | **34.83** | **87.17** | **79.01** | **80.68** | 67.11 | 67.80 |

Table 18: REMS & CAE score (in %) for all reasoning strategies across all datasets using Llama3.170B. R stands for “Refactoring,” U for “Unified.”

## C Detailed Error Analysis

We conduct a detailed error analysis across six datasets to identify the primary failure modes in pipeline-based table QA. As shown in plot 2 evidence extraction errors dominate in most datasets, often occurring before reasoning or code execution can contribute. However, we also observe notable secondary errors particularly in reasoning (e.g., TAT-QA) and code (e.g., WikiTQ) which vary by dataset structure and modality.

## SEAR: Step 2

Your task is to comprehend the crucial steps for a given table and question, making each step more detailed and ensuring they effectively lead to the correct answer.

Few Examples are given below with detailed steps by elaborating from the crucial steps. Interpret the examples and understand the task to comprehend the crucial steps according to the specific table, text and question to reach the answer efficiently.

Table:

| Model | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013 |
|---|---|---|---|---|---|---|---|---|---|
| Škoda Octavia | 233322 | 270274 | 309951 | 344857 | 317335 | 349746 | 387200 | 409360 | 359600 |
| Škoda Fabia | 236698 | 243982 | 232890 | 246561 | 264173 | 229045 | 266800 | 255025 | 202000 |
| Škoda Superb | 22091 | 20989 | 20530 | 25645 | 44548 | 98873 | 116700 | 106847 | 94400 |
| Škoda Roomster | | 14422 | 66661 | 57467 | 47152 | 32332 | 36000 | 39249 | 33300 |
| Škoda Yeti | | | | | 11018 | 52604 | 70300 | 90952 | 82400 |
| Škoda Rapid | | | | | | | 1700 | 9292 | 103800 |
| Škoda Citigo | | | | | | | 509 | 36687 | 45200 |

Question: How many Skoda cars were sold in 2010?

Crucial Steps:

Identify Objective: Define the goal.
Evidence Extraction:Extract relevant rows, columns, and text.
Python Code Generation: Generate single Python code to sum the extracted values.

LLM Output

Detailed Steps:

Identify Objective: Determine that the goal is to find the total number of Škoda cars sold in 2010.
Evidence Extraction: Extract the relevant data (the 2010 column) for all models listed in the table.
Python Code Generation: Generate Python code to sum the extracted values for the year 2010.

Figure 4: Sear Step 2 Prompt Example

### Dataset-specific Observations

**HybridQA.** Most errors result from incorrect row/column selection, driven by surface-level matches to look-alike strings (e.g., “season,” “division”) and missed disambiguators (e.g., years or suffixes like “(q)”). These tokens frequently appear in adjacent cells or parentheses, making shallow matches more likely. A small number of reasoning errors stem from failure to disambiguate linked entities across tables. Code issues are rare due to minimal programmatic computation.

**HiTABs.** Models often fail due to header ambiguity and dense, repetitive tabular layouts. Errors stem from misidentified rows/columns and unaccounted qualifiers like ranges or footnotes. Since the task is primarily table lookup, downstream reasoning errors are minimal, and code is not a significant factor.

**MultiHiertt (Multi).** High error rate in evidence selection is attributed to multi-hop grounding and similar-looking headers across tables. Subtle distinctions in qualifiers or column semantics are frequently overlooked. The remaining errors are due to misinterpretation of multi-hop logic, where the model fails to chain intermediate inferences.

**TAT-QA.** While evidence errors are common, reasoning mistakes form a large minority (36%), often caused by temporal mismatches (e.g., Q1 vs. FY) or incorrect unit normalization (e.g., billions vs. millions). Models struggle to align period-based values or compute correct numerical operations even with correct evidence.

**FETAQA.** Evidence-level failures persist due to repeated phrases across seasons, clubs, and divisions. Parenthetical markers (e.g., “(q)”) in headers lead to grounding mismatches. Some errors stem from reasoning failures, particularly aggregation mismatches or improper scoping across semi-structured tables. A few errors involve minor code missteps, such as summing incorrect subsets.

**WikiTQ.** Unlike others, code generation errors dominate (44%). The model often produces incorrect filters or aggregation logic due to brittle parsing of semi-structured HTML-derived tables. Even when correct evidence is identified, the final output is wrong due to mis-joins, faulty parsing of footnotes, or wrong aggregation. Evidence errors (40%) and reasoning mistakes (16%) persist but are less frequent.

The analysis reveals that evidence selection remains the primary bottleneck across most datasets. However, reasoning errors are increasingly relevant in multi-hop or temporal computation tasks (TAT-QA), and code execution errors emerge as a major challenge in semi-structured, programmatic tasks like WikiTQ. Addressing early-stage grounding *and* late-stage execution together is critical for end-to-end accuracy.

## D DataSet Overview

1. **FeTaQA(Nan et al., 2021) :** A Wikipedia-based table QA dataset that requires generating long-form answers by integrating multiple discontinuous facts and reasoning across structured tables. **Temporal Questions: 1,582**
2. **FinQA(Chen et al., 2021) :** A financial QA dataset from reports, requiring expert-verified multi-step numerical reasoning and gold reasoning programs for explainability. **Temporal Questions: 962**
3. **HiTab(Cheng et al., 2022) :** A cross-domain QA and NLG dataset featuring hierarchical tables, analyst-authored questions, and fine-grained annotations for complex numerical reasoning. **Temporal Questions: 897**
4. **HybridQA(Chen et al., 2020b) :** A QA dataset requiring reasoning over Wikipedia tables and linked free-form text, demanding both tabular and textual data for accurate answers. **Temporal Questions: 1,528**
5. **MultiHierTT(Zhao et al., 2022) :** A financial QA benchmark requiring reasoning over multiple hierarchical tables and long unstructured text, with detailed multi-step numerical reasoning annotations. **Temporal Questions: 1,587**
6. **Squall(Shi et al., 2020) :** An extension of WikiTableQuestions with manually created SQL equivalents and fine-grained alignments, supporting structured query reasoning in tabular environments. **Temporal Questions: 774**
7. **TAT-QA(Zhu et al., 2021) :** A financial QA dataset requiring reasoning over both tabular and textual data, involving operations like arithmetic, counting, and sorting for quantitative and qualitative analysis. **Temporal Questions: 2,244**
8. **WikiTableQ(Pasupat and Liang, 2015) :** A Wikipedia-based QA dataset with trivia-style questions requiring factual and numerical reasoning over tables with at least 8 rows and 5 columns. **Temporal Questions: 1,504**

## SEAR: Step 3

You are responsible for delivering precise answers by strictly following the provided detailed steps. Each answer must be carefully reasoned, supported by clear explanations, and based on thorough analysis of the given table, text, and question.

Few examples are given below. Interpret the examples and understand the task to answer the tabular question according to the specific table, text, question by following the detailed steps to answer efficiently.

Table:

| Model | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013 |
|---|---|---|---|---|---|---|---|---|---|
| Škoda Octavia | 233322 | 270274 | 309951 | 344857 | 317335 | 349746 | 387200 | 409360 | 359600 |
| Škoda Fabia | 236698 | 243982 | 232890 | 246561 | 264173 | 229045 | 266800 | 255025 | 202000 |
| Škoda Superb | 22091 | 20989 | 20530 | 25645 | 44548 | 98873 | 116700 | 106847 | 94400 |
| Škoda Roomster | | 14422 | 66661 | 57467 | 47152 | 32332 | 36000 | 39249 | 33300 |
| Škoda Yeti | | | | | 11018 | 52604 | 70300 | 90952 | 82400 |
| Škoda Rapid | | | | | | | 1700 | 9292 | 103800 |
| Škoda Citigo | | | | | | | 509 | 36687 | 45200 |

Question: How many Skoda cars were sold in 2010?

Crucial Steps:

Identify Objective: Define the goal.
Evidence Extraction:Extract relevant rows, columns, and text.
Python Code Generation: Generate single Python code to sum the extracted values.

Detailed Steps:

Identify Objective: Determine that the goal is to find the total number of Škoda cars sold in 2010.
Evidence Extraction: Extract the relevant data (the 2010 column) for all models listed in the table.
Python Code Generation: Generate Python code to sum the extracted values for the year 2010.

LLM Output

Answer and Reasoning:

Identify Objective: The goal is to find the total number of Škoda cars sold in 2010.

Evidence Extraction: Extract the relevant data (the 2010 column) for all models listed in the table:
Škoda Octavia: 349746
Škoda Fabia: 229045
Škoda Superb: 98873
Škoda Roomster: 32332
Škoda Yeti: 52604
Škoda Rapid: N/A (No data for 2010)
Škoda Citigo: N/A (No data for 2010)

Python Code Generation: We can now sum the extracted values for the year 2010 using Python code
``` python ```
```
# Data for 2010
skoda_sales_2010 = [349746, 229045, 98873, 32332, 52604]
# Calculate the total sales for 2010
total_sales_2010 = sum(skoda_sales_2010)
print(f"Final Answer: {{total_sales_2010}}")
#Done
```
Answer and Reasoning: Summing these values gives the total number of Škoda cars sold in 2010.
Final Answer: 762600

Figure 5: Sear Step 3 Prompt Example

## SEAR_UNIFIED PROMPT

Instruction
You are a adaptive-reasoner with the capabilities to select or merge steps to create the most appropriate reasoning pathway based on the tabular question provided by the user. You can even develop new reasoning steps by combining the new steps or learning from illustrations to create new pathways depending on the provided problem.

Steps for Adaptive Reasoning:
Each section has multiple approaches, you do not have to use all the approaches. Understand their use-cases and then pick minimal relevant steps to create your own optimal approach to answer the question.

Problem Understanding:
- Determine the objective: Identify the goal or desired outcome of the reasoning process.
- Understand the problem: Comprehend the nature and scope of the problem.

Reasoning Process:
- Step-by-step reasoning: Approach the problem logically, ensuring clarity at each step or stage.
- Extract relevant information: Gather all necessary data and details pertinent to the problem, by extracting relevant rows, columns and textual information.
- Decomposition of problem into sub-problems: Break down the main question into smaller and more manageable sub questions.
- Individually answer each sub-problem with reasoning: Apply logical steps to solve each sub question separately.
- Write a single Python program for solving the problem: Create a detailed unified Python script with comments describing the steps and stages.
- Individually write a Python program for each sub-problem: Develop separate Python scripts for each sub-problem, ensuring modularity and clarity.

Conclusion:
- Summarize findings: Combine the results from each step or sub question to give the final answer as Final Answer: {{Answer}}.
- Combine Python code: If necessary, integrate the individual Python scripts into a cohesive program at the end. Print the final answer as Final Answer: {{Answer}}, end your code with a comment "#Done".

Error Detection:
- Review each step or sub-problem: Ensure each step or sub-problem has been addressed thoroughly and correctly.
- Ensure logical flow: Verify that the reasoning process flows logically from one step to the next.
- Check Python program for syntax and errors: Confirm that the final Python program is syntactically correct and free of errors.

**Helpful Tips for Creating Appropriate and Optimal Approach**:
- Understand what is asked in the question, mention all the steps required to answer the question and why each step is necessary.
- If the question can be broken into smaller and more manageable sub questions, always decompose the question into relevant sub questions.
- If there are **calculations involved you must use python code** for performing calculations and reaching the final answer.
- If the question is directly answerable by direct look up from the tabular data or from the extracted evidence then provide a direct answer.

Table:
Context:

### Race Results Overview

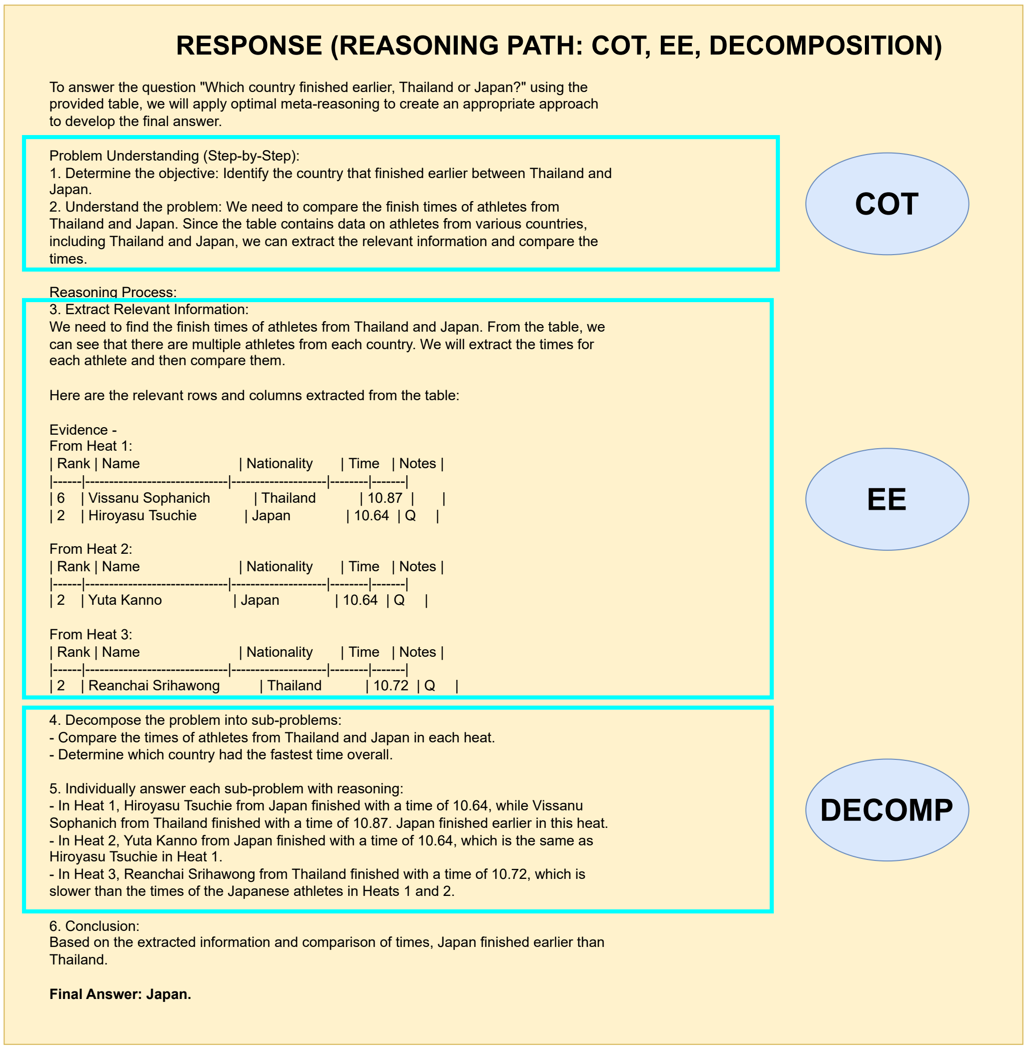
This table showcases the results of various athletes who participated in different heats, including their times and nationalities.

| Rank | Heat | Name | Nationality | Time | Notes |
|------|------|------|------|------|------|
| 1 | 1 | Salem Al-Yami | Saudi Arabia | 10.55 | Q |
| 2 | 1 | Hiroyasu Tsuchie | Japan | 10.64 | Q |
| 3 | 1 | Khaled Yousef Al-Obaidli | Qatar | 10.68 | Q |
| 4 | 1 | Chintake De Zoysa | Sri Lanka | 10.78 | q |
| 5 | 1 | Suminda Mendis | Sri Lanka | 10.82 | q, PB |
| 6 | 1 | Vissanu Sophanich | Thailand | 10.87 | |
| 1 | 2 | Gennadiy Chernovol | Kazakhstan | 10.59 | Q |
| 2 | 2 | Yuta Kanno | Japan | 10.64 | Q |
| 3 | 2 | Shen Yunbao | China | 10.72 | Q |
| 4 | 2 | Tsai Meng-Lin | Chinese Taipei | 10.74 | q |
| 5 | 2 | Tan Kok Lim | Malaysia | 10.83 | q |
| 6 | 2 | Ahmad Hudeib Al-Mamari | Oman | 10.97 | |
| 1 | 3 | Jamal Al-Saffar | Saudi Arabia | 10.57 | Q |
| 2 | 3 | Reanchai Srihawong | Thailand | 10.72 | Q |
| 3 | 3 | Shin Jung-Ki | South Korea | 10.79 | [2] |
| 4 | 3 | Chen Tien-Wen | Chinese Taipei | 10.74 | q |
| 5 | 3 | To Wai Lok | Hong Kong | 10.92 | |
| 6 | 3 | Poh Seng Song | Singapore | 11.10 | SB |
| 1 | 4 | Chen Haijian | China | 10.65 | Q |
| 2 | 4 | Saad Faraj Al-Shahwani | Qatar | 10.67 | Q |
| 3 | 4 | Azmi Ibrahim | Malaysia | 10.78 | Q |
| 4 | 4 | Chiang Wai Hung | Malaysia | 10.89 | |
| 5 | 4 | Nguyen Thanh Hai | Vietnam | 11.16 | PB |
| 6 | 4 | Piphop Rasme Prum Keo | Cambodia | 11.70 | PB |

Question : which country finished earlier, thailand or japan?

Optimal Approach:

Figure 6: SEAR_UNIFIED Prompt Example


COT
EE
DECOMP

**RESPONSE (REASONING PATH: COT, EE, DECOMPOSITION)**

To answer the question "Which country finished earlier, Thailand or Japan?" using the provided table, we will apply optimal meta-reasoning to create an appropriate approach to develop the final answer.

Problem Understanding (Step-by-Step):
1. Determine the objective: Identify the country that finished earlier between Thailand and Japan.
2. Understand the problem: We need to compare the finish times of athletes from Thailand and Japan. Since the table contains data on athletes from various countries, including Thailand and Japan, we can extract the relevant information and compare the times.

Reasoning Process:
3. Extract Relevant Information:
We need to find the finish times of athletes from Thailand and Japan. From the table, we can see that there are multiple athletes from each country. We will extract the times for each athlete and then compare them.

Here are the relevant rows and columns extracted from the table:

```
Evidence -
From Heat 1:
| Rank | Name                          | Nationality        | Time   | Notes |
|------|-------------------------------|--------------------|--------|-------|
| 6    | Vissanu Sophanich             | Thailand           | 10.87  |       |
| 2    | Hiroyasu Tsuchie              | Japan              | 10.64  | Q     |

From Heat 2:
| Rank | Name                          | Nationality        | Time   | Notes |
|------|-------------------------------|--------------------|--------|-------|
| 2    | Yuta Kanno                    | Japan              | 10.64  | Q     |

From Heat 3:
| Rank | Name                          | Nationality        | Time   | Notes |
|------|-------------------------------|--------------------|--------|-------|
| 2    | Reanchai Srihawong            | Thailand           | 10.72  | Q     |
```

4. Decompose the problem into sub-problems:
- Compare the times of athletes from Thailand and Japan in each heat.
- Determine which country had the fastest time overall.

5. Individually answer each sub-problem with reasoning:
- In Heat 1, Hiroyasu Tsuchie from Japan finished with a time of 10.64, while Vissanu Sophanich from Thailand finished with a time of 10.87. Japan finished earlier in this heat.
- In Heat 2, Yuta Kanno from Japan finished with a time of 10.64, which is the same as Hiroyasu Tsuchie in Heat 1.
- In Heat 3, Reanchai Srihawong from Thailand finished with a time of 10.72, which is slower than the times of the Japanese athletes in Heats 1 and 2.

6. Conclusion:
Based on the extracted information and comparison of times, Japan finished earlier than Thailand.

**Final Answer: Japan.**

Figure 7: The figure illustrates the response path followed by SEAR_Unified Prompting. The reference prompt is provided in Figure 6

**Input:**

You are an expert LLM evaluator tasked with assessing the accuracy of model responses against gold standard answers. Your role is to determine if the core content and intent of the model's response align with the gold answer, considering various answer formats and implicit information.

**Key Guidelines**

- **Understand the question's essence,** including specific operations or units mentioned.
- **Compare model responses** to gold answers, focusing on key information.
- **Allow a small margin of error** (±0.1%) for numerical answers.
- **Recognize correct answers in different formats**, such as percentages and decimals.
- **Consider implicit information and context** in responses.
- **For list-type answers:**
  - Evaluate based on content rather than order.
  - If more than **two elements are missing** (context-dependent), evaluate as incorrect.
- **Assess mathematical answers** based on value range unless a specific value is required.
- **Check for appropriate units** in mathematical answers.

**Final Judgment**
Provide a "**Yes**" or "**No**" judgment without explanation unless explicitly requested

Figure 8: Prompt for Contexual Answer Evaluation(CAV)

## Input:

**Instruction**

You are given the following **Question** and **Context**. The Context includes a table that may be incomplete, ambiguous, or poorly structured. Your task is to produce a **cleaned version of the table** that improves its clarity and structure so that it can be correctly used to answer the Question.

**Guidelines**

1. **Do not add, remove, or alter any data.** Only restructure and clarify what is already present.
2. You may improve the **table title** if it is missing or ambiguous:
   - If a title is missing, infer an appropriate one based on the **question** and table content.
   - If the existing title is unclear or misleading, revise it for clarity while keeping its original meaning.
3. You may improve the **table headers** if needed:
   1. Rename ambiguous column/row headers for clarity.
   2. Ensure column and row labels accurately describe their content.
4. You may fix **structural inconsistencies**:
   1. Align misaligned data properly under the correct headers.
   2. Ensure row and column structures are uniform.
   3. Remove redundant headers or merge split headers where necessary.
5. The data should be kept in the same order whenever possible. However, if **minor reordering of rows or columns** helps fix structural issues, you may do so **only if it does not change or omit any data**.

**Output Format**

- Provide only the **cleaned table** as your output in a structured format appropriate for the data in **Markdown format**.
- **Do not add any explanations, reasoning, or commentary.**

**Question:** {question}
**Context:** {context}
**Now produce just the cleaned table**

Figure 9: Prompt for Refactoring Tables.

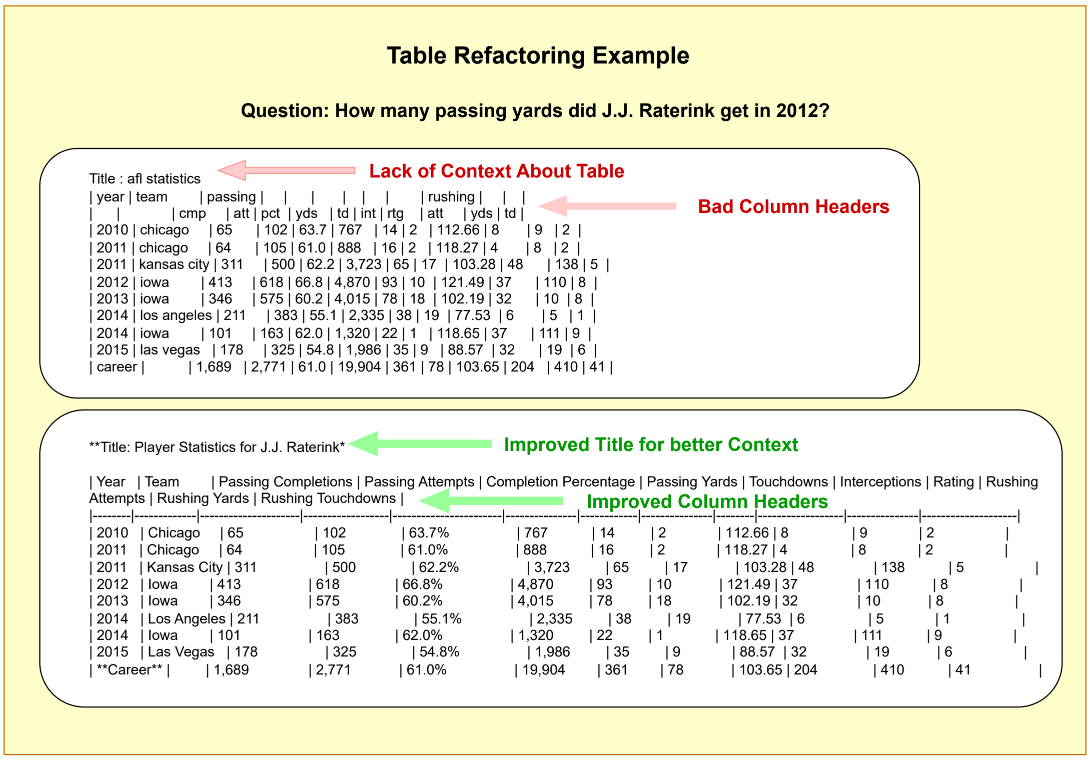


Table Refactoring Example
Question: How many passing yards did J.J. Raterink get in 2012?
Lack of Context About Table
Bad Column Headers
Improved Title for better Context
Improved Column Headers

```
Title : afl statistics
| year | team        | passing |     |      |       |    |     |       | rushing |     |    |
|      |             | cmp     | att | pct  | yds   | td | int | rtg   | att     | yds | td |
| 2010 | chicago     | 65      | 102 | 63.7 | 767   | 14 | 2   | 112.66 | 8      | 9   | 2  |
| 2011 | chicago     | 64      | 105 | 61.0 | 888   | 16 | 2   | 118.27 | 4      | 8   | 2  |
| 2011 | kansas city | 311     | 500 | 62.2 | 3,723 | 65 | 17  | 103.28 | 48     | 138 | 5  |
| 2012 | iowa        | 413     | 618 | 66.8 | 4,870 | 93 | 10  | 121.49 | 37     | 110 | 8  |
| 2013 | iowa        | 346     | 575 | 60.2 | 4,015 | 78 | 18  | 102.19 | 32     | 10  | 8  |
| 2014 | los angeles | 211     | 383 | 55.1 | 2,335 | 38 | 19  | 77.53  | 6      | 5   | 1  |
| 2014 | iowa        | 101     | 163 | 62.0 | 1,320 | 22 | 1   | 118.65 | 37     | 111 | 9  |
| 2015 | las vegas   | 178     | 325 | 54.8 | 1,986 | 35 | 9   | 88.57  | 32     | 19  | 6  |
| career |           | 1,689   | 2,771 | 61.0 | 19,904 | 361 | 78 | 103.65 | 204 | 410 | 41 |
```

```
**Title: Player Statistics for J.J. Raterink*

| Year   | Team        | Passing Completions | Passing Attempts | Completion Percentage | Passing Yards | Touchdowns | Interceptions | Rating | Rushing
Attempts | Rushing Yards | Rushing Touchdowns |
|--------|-------------|---------------------|------------------|-----------------------|---------------|------------|---------------|--------|-----------------|--------------|--------------------|
| 2010   | Chicago     | 65                  | 102              | 63.7%                 | 767           | 14         | 2             | 112.66 | 8               | 9            | 2                  |
| 2011   | Chicago     | 64                  | 105              | 61.0%                 | 888           | 16         | 2             | 118.27 | 4               | 8            | 2                  |
| 2011   | Kansas City | 311                 | 500              | 62.2%                 | 3,723         | 65         | 17            | 103.28 | 48              | 138          | 5                  |
| 2012   | Iowa        | 413                 | 618              | 66.8%                 | 4,870         | 93         | 10            | 121.49 | 37              | 110          | 8                  |
| 2013   | Iowa        | 346                 | 575              | 60.2%                 | 4,015         | 78         | 18            | 102.19 | 32              | 10           | 8                  |
| 2014   | Los Angeles | 211                 | 383              | 55.1%                 | 2,335         | 38         | 19            | 77.53  | 6               | 5            | 1                  |
| 2014   | Iowa        | 101                 | 163              | 62.0%                 | 1,320         | 22         | 1             | 118.65 | 37              | 111          | 9                  |
| 2015   | Las Vegas   | 178                 | 325              | 54.8%                 | 1,986         | 35         | 9             | 88.57  | 32              | 19           | 6                  |
| **Career** |         | 1,689               | 2,771            | 61.0%                 | 19,904        | 361        | 78            | 103.65 | 204             | 410          | 41                 |
```

Figure 10: Refactored Table Example