Table-r1: Self-Supervised and Reinforcement Learning for Program-Based Table Reasoning in Small Language Models

Rihui Jin , Zheyu Xin , Xing Xie , Zuoyi Li , Guilin Qi * , Yongrui Chen , Xingbang Dai , Tongtong Wu , Gholamreza Haffari

Posted Date: 6 June 2025

doi: 10.20944/preprints202506.0552.v1

Keywords: table reasoning; large language models; reinforcement learning; self-supervised learning

 doi:10.20944/preprints202506.0552.v1

Article

Table-r1: Self-Supervised and Reinforcement Learning for Program-Based Table Reasoning in Small Language Models

Rihui Jin [1,2], Zheyu Xin [1], Xing Xie [1], Zuoyi Li [1], Guilin Qi [1,*], Yongrui Chen [1], Xinbang Dai [1], Tongtong Wu [2], Gholamreza Haffari [2]

[1] Southeast University, Nanjing, China
[2] Monash University, Australia
* Correspondence: qgi@seu.edu.cn

Abstract: Table reasoning (TR) requires structured reasoning over semi-structured tabular data and remains challenging, particularly for small language models (SLMs, e.g., LLaMA-8B) due to their limited capacity compared to large LMs (LLMs, e.g., GPT-4o). To narrow this gap, we explore program-based TR (P-TR), which circumvents key limitations of text-based TR (T-TR)—notably in numerical reasoning—by generating executable programs. However, applying P-TR to SLMs introduces two challenges: (i) vulnerability to heterogeneity in table layouts, and (ii) inconsistency in reasoning due to limited code generation capability. We propose *Table-r1*, a two-stage P-TR method designed for SLMs. Stage 1 introduces an innovative self-supervised learning task, Layout Transformation Inference, to improve tabular layout generalization from a programmatic view. Stage 2 adopts a mix-paradigm variant of Group Relative Policy Optimization, enhancing P-TR consistency while allowing dynamic fallback to T-TR when needed. Experiments on four TR benchmarks demonstrate that *Table-r1* outperforms all SLM-based methods, achieving at least a 15% accuracy improvement over the base model (LLaMA-8B) across all datasets and reaching performance competitive with LLMs.

Keywords: table reasoning; large language models; reinforcement learning; self-supervised learning

1. Introduction

Table reasoning (TR) [1,2] is a fundamental yet challenging task, requiring structured reasoning over the two-dimensional semantics of tabular data [3]. With the emergence of large language models (LLMs), TR has seen notable progress [4]. Existing LLM-based methods fall into two main paradigms: text-based TR (T-TR), e.g., Table-CoT [5], which treats tables as plain text and generates answers directly; and program-based TR (P-TR), e.g., Binder [6], which generates executable programs to derive answers. Subsequent studies [7–9] have further advanced both approaches.[1]

However, when moving from LLMs (e.g., GPT-4o) to small language models (SLMs, e.g., Qwen-7B) [10], performance drops significantly (Figure 1(c)), primarily due to SLMs' weaker reasoning and text comprehension abilities. While prior efforts [11–13] have fine-tuned SLMs under the T-TR setting, these face inherent limits such as restricted context windows and poor numerical handling [14]. In contrast, P-TR methods naturally mitigate these issues [12,15][2]. This motivates a central question: **Can we design a P-TR method enabling SLMs to match or even surpass LLM performance?**

[1] Source code will be shown in https://github.com/AriKing11/Table_r1_public
[2] Further discussed in §5.3

 doi:10.20944/preprints202506.0552.v1

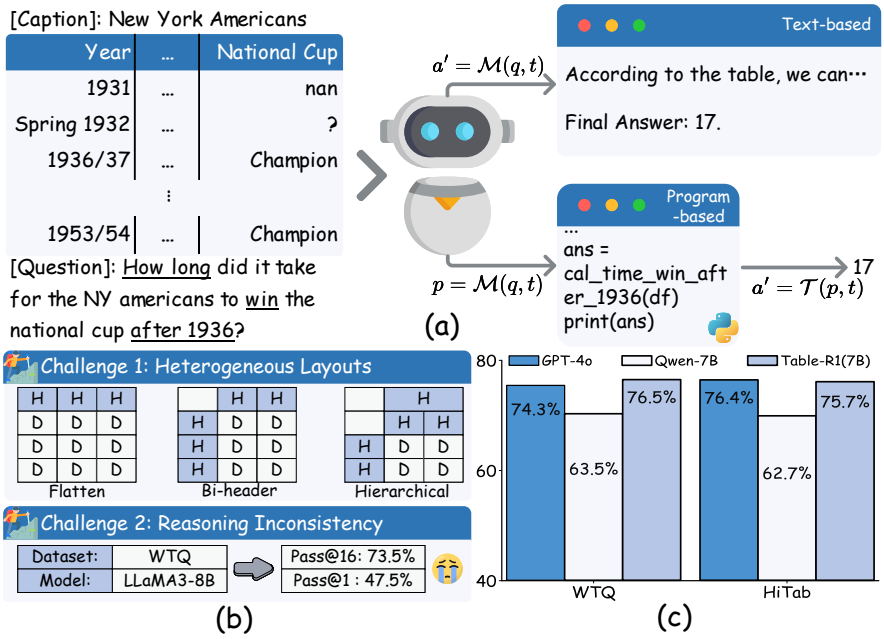

Figure 1. (a) Two TR paradigms: the text-based directly generates answers, while the program-based produces executable code (e.g., Python) to derive answers via a code executor. (b) Two challenges for SLMs in P-TR: heterogeneous layouts (e.g., complex and flexible header structures) hinder accurate indexing, and limited reasoning ability causes inconsistent performance. (c) Performance comparison on two TR benchmarks: LLMs(GPT-4o) outperform SLMs (Qwen-7B) by a large margin; *Table-r1*, built on Qwen-7B, significantly narrows the gap and rivals LLM performance.

Pursuing this direction, however, requires overcoming two key challenges of SLMs under the P-TR paradigm, as highlighted in Figure 1(b): (i) poor generalization to heterogeneous layouts (flat, bi-level, hierarchical headers), which impairs accurate referencing of contents via headers as arguments in programs; and (ii) reasoning inconsistency, as evidenced by high pass@16 (73.5%) but low pass@1 (47.5%) in single-shot settings. To address them, we propose *Table-r1*, a two-stage P-TR method designed for SLMs: In Stage 1, we introduce an innovative self-supervised learning task—Layout Transformation Inference—to enhance layout generalization by training the model to infer structural transformations from a programmatic perspective. This promotes accurate header usage and content referencing. In Stage 2, we adopt and extend Group Relative Policy Optimization (GRPO) [16] into a mix-paradigm variant that prioritizes P-TR while selectively leveraging T-TR as a fallback. The model dynamically switches strategies based on the context and prior reasoning trace, guided by several TR-specific rewards. GRPO then optimizes the model's policy toward higher-reward completions, enhancing both robustness and adaptability. Extensive experiments across four public TR benchmarks demonstrate that *Table-r1* consistently surpasses all prior SLM-based methods and achieves performance on par with LLMs.

The contributions of this paper can be summarized as follows:

1. We introduce an innovative self-supervised learning task tailored for the P-TR, designed to improve SLMs' ability to understand heterogeneous table layouts without manual annotation.
2. To the best of our knowledge, we are the first to apply GRPO to TR. We further adapt it into a mix-paradigm GRPO, design task-specific reward functions, and conduct an in-depth analysis of GRPO's effectiveness in the TR.
3. Extensive experiments demonstrate that *Table-r1* consistently outperforms all existing SLM-based methods and achieves performance competitive with that of LLMs across multiple TR benchmarks.

 doi:10.20944/preprints202506.0552.v1

2. Related Work

2.1. LM-based Table Reasoning

Recent advances in LM-based TR follow two main paradigms—Text-based TR and Program-based TR—as illustrated in Figure 1(a) and detailed in §3.1.

Text-based Paradigm. In this paradigm, the LLM takes the table and the corresponding question as input and directly generates the final answer. Chen [5] first demonstrated the potential of LLMs in TR by leveraging COT prompting [17] and ICL [18]. Building on this, subsequent works [19–21] introduced techniques such as parse-tree decoding, the ReAct [22], and format-aware table transformations to improve reasoning performance. However, such prompt-based methods struggle to generalize to SLMs, whose capabilities are significantly weaker than those of LLMs. Although several studies [7,11,13,23–25] have explored instruction tuning to bridge this gap, the performance of SLMs remains limited, primarily due to context length constraints. Truncated tables often omit critical information required for accurate reasoning. Motivated by these limitations, our work takes an alternative direction to unlock the potential of SLMs better, enabling them to excel in TR tasks.

Program-based Paradigm. This paradigm prompts the LLM to generate executable code, typically in Python, based on the provided table and question, and subsequently executes the code to derive the final answer. Binder [6] was the first to introduce the Program-of-Thought framework [14,26] to TR tasks. These methods leverage an external code executor to overcome limitations in context length and numerical reasoning commonly observed in LLMs [27,28]. As a result, P-TR methods have gained substantial traction, with follow-up works [8,29–33] integrating techniques such as ReAct, Tree-of-Thoughts [34], and graph-based modeling to enhance reasoning capabilities further. Notably, some methods [15,35] have adopted P-TR as the primary framework, with text-based paradigms playing a supplementary role to improve robustness and coverage. However, most of these methods remain heavily dependent on the code generation abilities of LLMs, posing significant challenges when adapting to SLMs. Although some initial efforts [12,33] have explored P-TR in the context of small models, a considerable performance gap still exists. In this work, we build upon the P-TR paradigm and propose an innovative method that enables SLMs to achieve performance comparable to that of their larger counterparts.

2.2. Reinforcement Learning for LLM-based Reasoning

Recently, a growing body of research has focused on enhancing the reasoning capabilities of LLMs. The impressive performance of o1 [36] in complex reasoning tasks has spurred the academic community to focus on optimizing models through RL. Notably, DeepSeek-R1 [37] and Kimi-1.5 [38] have both utilized advanced versions of Proximal Policy Optimization [39], resulting in significant improvements in LLM performance on verifiable tasks, such as mathematics and code generation. Among these, the GRPO [16] algorithm employed by DeepSeek-R1 has been widely validated for its effectiveness in boosting the reasoning abilities of SLMs [40–42]. Despite these advancements, to date, no work has explicitly applied RL-based methods to the domain of TR.

3. Preliminaries

3.1. Problem Statement

In the TR task, each data sample is formalized as a triplet $\langle t, q, a \rangle$, where $t \in \mathbb{T}$ denotes a semi-structured table, $q \in \mathbb{Q}$ represents a natural language question, and $a \in \mathbb{A}$ is the ground-truth answer derived from table t. A semi-structured table t is composed of a header H and a body section D. The header H encodes schema-level semantics and is designed for readability. Unlike flat database schemas, it may have a hierarchical layout or be split into top and left headers, especially in financial reports [43,44]. The goal of models in TR is to learn a mapping function $\phi : \mathbb{Q} \times \mathbb{T} \rightarrow \mathbb{A}$, which outputs a correct answer a' based on the input pair (q, t). In T-TR, the reasoning process is directly handled by a language model $\mathcal{M}$, which predicts the answer in a single step:

$$a' = \mathcal{M}(q, t), \tag{1}$$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 June 2025 doi:10.20944/preprints202506.0552.v1

where a' is the model's generated response. Alternatively, P-TR paradigm decomposes the TR into two stages: the model $\mathcal{M}$ first produces an intermediate program p, which is then executed by an external executor $\mathcal{E}$ to obtain the final answer:

$$p = \mathcal{M}(q, t), \quad a' = \mathcal{E}(p, t). \tag{2}$$

In our method, the program p corresponds to Python code.

4. Methodology

We propose *Table-r1*, a two-stage training method for enhancing TR. As illustrated in Figure 2, Stage 1 (§4.1) strengthens the model's understanding of tabular layouts and semantics; Stage 2 (§4.2) enhances code-based reasoning via reinforcement learning.

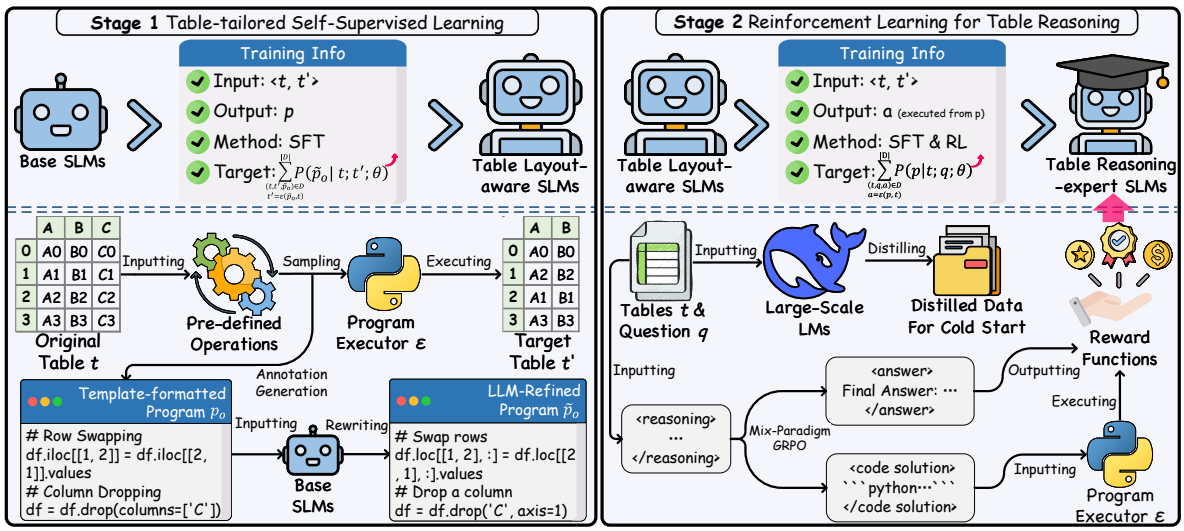

Figure 2. Overview of *Table-r1*. Stage 1 introduces an SSL task to improve layout understanding, with annotations auto-generated (bottom-left). Stage 2 applies RL, starting from teacher-guided cold start and followed by a mix-paradigm GRPO, enabling dynamic strategy selection. Reward functions provide numerical feedback based on outputs.

4.1. Stage 1: Self-Supervised Learning Tailored for Program-based Table Reasoning

Real-world tables often deviate from standard formats, exhibiting diverse layouts—e.g., flat, hierarchical, bi-directional headers—that hinder semantic parsing and accurate indexing.

To mitigate this issue, we propose a self-supervised task (SSL) task called Layout Transformation Inference (LTI), which trains the model to compare two tables—an original table t and its transformed counterpart t'—and generate a program $\tilde{p}_o$ that reconstructs t' from t:

$$t' = \mathcal{E}(\tilde{p}_o, t), \tag{3}$$

where $\mathcal{E}$ is a program executor. LTI demands fine-grained comparison between t and t' to identify layout changes and express them via executable transformations. This encourages the model to distinguish headers from data and use headers as structural anchors. The task is non-trivial—GPT-4o achieves only 45% accuracy in ICL, highlighting the difficulty and potential for improving the model's understanding ability of layouts without manual annotation.

4.1.1. Training Data Synthesis

Synthetic Instance Construction To automatically generate target tables, we define a set of transformation operations $\mathcal{O} = \{o_1, o_2, \ldots, o_K\}$, where each o_i is a deterministic function modifying

 doi:10.20944/preprints202506.0552.v1

the layout of a table t (e.g., row swapping, column deletion; see Appendix D for details). For each t, we apply a random sequence of n operations ($1 \leq n \leq 3$) to generate a transformed table t':

$$p_o = o_n \circ \cdots \circ o_1, \quad t' = \mathcal{E}(p_o, t), \tag{4}$$

where p_o represents the composition of applied operations and serves as the supervision labels.

LLM-Based Label Rewriting Directly using template-based programs p_o in SSL can introduce distributional mismatch, causing catastrophic forgetting [45]. To mitigate this, we adopt a continual learning strategy based on self-distillation [46]. Specifically, the base model $\mathcal{M}_{\text{base}}$ rewrites the raw synthetic program p_o into a semantically equivalent but more fluent version $\tilde{p}_o$:

$$\tilde{p}_o = \mathcal{M}_{\text{base}}(t, t', p_o), \tag{5}$$

which serves as the final training label.

4.1.2. Fine-Tuning Procedure

We fine-tune the model on synthesized triplets $(t, t', \tilde{p}o)$ using the following objective:

$$\mathcal{L}_{\text{LTI}} = \frac{1}{|D|} \sum_{\substack{(t,t',\tilde{p}_o)\in D \\ \mathcal{E}(\tilde{p}_o,t)=t'}} \left[-\log P(\tilde{p}_o \mid t; t'; \theta)\right], \tag{6}$$

where D is the training set and θ the model parameters. The loss is computed only when $\tilde{p}_o$ faithfully transforms t into t'.

4.2. Stage 2: Reinforcement Learning for Table Reasoning

We enhance code-based reasoning via RL. GRPO [37] is particularly well-suited for SLMs due to its efficiency and strong performance on verifiable tasks, outperforming traditional methods such as PPO [39]. Meanwhile, prior work [15,35] shows that T-TR can complement P-TR under specific contexts. Motivated by these insights, we integrate a mix-paradigm variant of GRPO into *Table-r1*, enhancing P-TR consistency while allowing dynamic fallback to T-TR when needed. We now detail the components: completion templates (§4.2.1), distillation-based cold start (§4.2.2), reward functions (§4.2.3), and the mix-paradigm GRPO (§4.2.4).

4.2.1. Completion Templates

As shown in Figure 2, we adopt two structured output formats. Both prompt the model to generate intermediate reasoning before deciding on an answer format. If the P-TR template is selected, the answer is enclosed in `<code_solution>` tags containing Python code; otherwise, a direct answer appears within `<answer>` tags. This design, inspired by [47], encourages deliberate reasoning and supports flexible strategy selection. Examples are provided in Appendix E.3.

4.2.2. Cold Start with Distilled Data

Inspired by [40], we adopt a distillation-based cold start strategy before RL. We employ LLMs [48] as teacher models to generate reasoning traces and code solutions on the training split, following the prompt format in Appendix E.2. The distilled data is then used to perform SFT on the base model.

4.2.3. Reward Functions

Reward design is central to RL [40]. In GRPO, reward functions evaluate each model-generated completion, guiding optimization toward better outputs. Formally, we define:

$$R(c_i) = \sum_j R_j(c_i, t, a), \tag{7}$$

 doi:10.20944/preprints202506.0552.v1

where R_j denotes the j-th reward component, and $R(c_i)$ is the total reward for completion c_i.

Beyond format-adherence reward functions (see Table A2 for more details), the following reward functions R_j are introduced to encourage more reliable and interpretable generation:

- **Compilation Correctness.** Program-based outputs receive the reward only when the code compiles successfully. Text-based outputs are assumed valid by default, which introduces a bias that is counteracted by other reward terms.
- **Answer Correctness.** For P-TR outputs, we execute the generated code to obtain the answers; for text-based outputs, we use regular expression matching. To refine reward granularity, we compare model outputs with golden answers using similarity metrics. Incorrect but compilable outputs (e.g., `print(final answer)`) are penalized heavily. We also penalize the use of text-based answers on questions requiring P-TR (e.g., extremum queries over truncated tables).
- **Short Code Penalty.** To discourage degenerate behaviors (e.g., overly short programs), we impose penalties on code completions that are below a specified length threshold.

4.2.4. Mix-paradigm GRPO

We extend GRPO to a *mix-paradigm* variant, allowing the model to prioritize P-TR while flexibly falling back to T-TR when appropriate. GRPO optimizes a policy by maximizing the relative advantage of high-reward completions, regularized against a reference policy. Given a query-table pair $\langle q, t \rangle$, the model generates multiple completions $c_i = [s; z']$, where s is a reasoning trace and z' is the final answer in a specific format. We define the output space $\mathcal{Z} =$ program, text, where each $z \in \mathcal{Z}$ denotes a distinct answer form—executable code or direct output. The model implicitly selects z' during generation, conditioned on s:

$$z' = \arg\max_{z \in \mathcal{Z}} P(z \mid s; \theta). \tag{8}$$

Each c_i is then evaluated using a TR-specific reward function R_{c_i} (defined in §4.2.3). We compute its relative advantage within the batch as:

$$A(c_i) = \frac{R(c_i) - \mu_R}{\sigma_R + \epsilon}, \tag{9}$$

where μ_R and σ_R are the mean and standard deviation of rewards among the sampled completions c_i in the same batch, and ϵ is a small constant for numerical stability. The policy is updated by maximizing a clipped surrogate objective:

$$\mathcal{L}_{\text{GRPO}} = \mathbb{E}_{c_i \sim \pi_\theta}[\min(r(c_i)A(c_i),\ \text{clip}(r(c_i), 1-\epsilon, 1+\epsilon)A(c_i))] - \lambda \cdot \text{KL}(\pi_\theta || \pi_{\theta_{\text{ref}}}), \tag{10}$$

where $r(c_i) = \pi_\theta(c_i) / \pi_{\theta_{\text{ref}}}(c_i)$ is the likelihood ratio, and λ controls the KL regularization strength. This encourages exploration of higher-reward answers while constraining divergence from the reference policy.

5. Experiments

In this section, we first present the datasets and implementation details. We then report experimental results comparing our method against strong baselines, including SOTA, to validate its effectiveness. Additionally, we explore several key findings that arise from our study, which we formulate into the following research questions:

- **RQ1**: For SLMs, what are the respective strengths and weaknesses of the T-TR and P-TR paradigms in handling real-world TR tasks?
- **RQ2**: How does each module contribute to the overall performance gains in TR, and in which scenarios does mix-paradigm GRPO demonstrate clear advantages?
- **RQ3**: What are the respective roles and impacts of SFT and RL in improving the reasoning capability of P-TR models?

 doi:10.20944/preprints202506.0552.v1

- **RQ4**: To what extent does organizing training data by difficulty levels influence the learning efficiency and final performance of GRPO?

5.1. Experimental Settings

Datasets & Metrics & Implementation. We evaluate on four widely used TR benchmarks. Dataset statistics are summarized in Table 1. For all datasets except AIT-QA, we use the original train-test splits; AIT-QA is randomly split 8:2 based on questions. Following prior work [6,15], we adopt self-consistency (SC) with majority voting, using 5 samples per prediction, and report exact match accuracy as the evaluation metric. Implementation details can be found in Appendix C.

Table 1. Dataset statistics summary

Dataset	Header	# t	# q	Domain
WTQ [49]	Flat	2.1k	22k	Geneal
TabFact [50]	Flat	16k	118k	General
HiTab [43]	Hierarchical	3.5k	10k	Crime, Health
AIT-QA [44]	Hierarchical	116	515	Airline

Baselines. We compare against recent TR methods (see Table 2), covering both text-based and program-based paradigms. Additionally, we evaluate the text-based and program-based performance of SOTA LLMs (GPT-4o [51], DeepSeek-v3 [48], DeepSeek-R1 [37]) and base models (Qwen2.5-Coder-7B [52], LLaMA3.1-8B-Instruct [53]) under the same prompt settings with SC=5 as in *Table-r1*.

Table 2. Main results of LLM-based methods across various TR datasets. **Paradigm** indicates the modeling style: `T-TR` (text-based) and `P-TR` (program-based). The evaluation metric is accuracy. The E2E (one-shot) series share the same prompts as Table-R1, with one-shot demonstration included. Averaged over three runs; ± denotes standard deviation.

Method	Base Model	Paradigm	Flatten Tables		Hierarchical Tables	
			WTQ	TabFact	HiTab	AIT-QA
Large-Scale LLMs						
E2E (one-shot)	GPT-4o	`T-TR`	63.2	87.6	82.4	88.5
E2E (one-shot)	DeepSeek-v3 (670B)	`T-TR`	61.0	84.5	**85.3**	**91.7**
E2E (one-shot)	GPT-4o	`P-TR`	74.3	88.1	76.4	86.8
E2E (one-shot)	DeepSeek-v3 (670B)	`P-TR`	75.0	87.2	77.0	87.0
E2E (one-shot)	DeepSeek-r1 (670B)	`P-TR`	**82.0**	**93.0**	82.3	90.0
TableParser [54]	Qwen2-72B-Inst	`T-TR`	–	–	44.6	64.9
Table-Critic [55]	Qwen2-72B-Inst	`T-TR`	77.2	92.6	–	–
GraphOTTER [9]	Qwen2-72B-Inst	`P-TR`	–	–	73.7	88.3
PoTable [56]	LLaMA3.1-70B-Inst	`P-TR`	65.6	87.1	–	–
TableMaster [57]	LLaMA3.1-70B-Inst	`P-TR`	78.0	91.2	–	–
E5 [58]	GPT-4	`P-TR`	65.5	88.8	85.1	–
Norm-DP&Agent [15]	GPT-3.5	`P-TR`	73.7	88.5	–	–
TIDE-DP&Agent [35]	GPT-3.5	`P-TR`	75.0	89.8	–	–
Small-Scale LLMs						
E2E (one-shot)	LLaMA3.1-8B-Inst	`P-TR`	55.3	65.2	49.2	48.8
E2E (one-shot)	Qwen2.5-Coder-7B-Inst	`P-TR`	63.5	76.0	62.7	55.5
TableLLama [11]	LLaMA2-7B	`T-TR`	–	82.6	60.5	–
TAMA [7]	LLaMA3.1-8B-Inst	`T-TR`	52.9	73.8	63.5	89.2
TableGPT2 [13]	Qwen2.5-7B	`T-TR`	61.42	77.8	70.3	–
TableLoRA [25]	LLaMA3-8B-Inst	`T-TR`	53.5	84.0	58.6	–
Structlm [24]	Mistral-7B	`T-TR`	56.8	84.6	–	–
Tart [12]	DeepSeek-7B	`P-TR`	–	71.3	–	–
Table-r1	Qwen2.5-Coder-7B-Inst	`P-TR`	**76.5** ± 0.74	**85.3** ± 0.81	**76.7** ± 0.72	**89.9** ± 0.88
Table-r1	LLaMA3.1-8B-Inst	`P-TR`	74.3 ± 0.77	84.4 ± 0.85	74.2 ± 1.03	86.5 ± 0.93

 doi:10.20944/preprints202506.0552.v1

5.2. Overall Performance

Table 2 reports results of Table-R1 against all baselines across four TR benchmarks. Our method consistently outperforms all SLMs of comparable scale and matches the performance of top-tier LLMs. Several key observations emerge: (i) For LLMs, P-TR performs better on flattened tables (e.g., WTQ, TabFact), whereas T-TR is more effective on hierarchical tables (e.g., HiTab, AIT-QA); see § 5.3 for analysis. (ii) Models like Table-R1 and v3, which operate without external plugins, achieve the best results—highlighting the critical role of the base model's capacity in TR. That said, proper strategies still provide substantial gains, even for smaller models.

5.3. Error Studies (RQ1)

To analyze the limitations of the two paradigms (RQ1), we manually inspect 200 instance outputs each from WTQ and HiTab using Qwen2.5-7B-Instruct (text-based) and Qwen2.5-Coder-7B-Instruct (program-based). The program-based model is restricted to the first 10 rows due to token limits, while the text-based model processes the entire table (truncated if exceeding 4096 tokens). We follow the classification scheme in [15,58], assigning each prediction to one of six error categories:

- Missing Table Information (MTI): The answer-relevant content is truncated and unavailable to the model.
- Information Extraction Errors (IEE): Caused by structural heterogeneity, manifesting as hallucinations or index misuse (Examples shown in Figure A4 and Figure A5).
- Computation Errors (CMP): Errors stemming from incorrect numerical reasoning.
- Code Syntax Errors (CSE): The generated code fails to compile due to syntactic issues.
- Logical Errors in Code (CLE): The code compiles and runs but yields incorrect results due to flawed reasoning or filtering logic. (An example shown in Figure A6).
- Answer Misalignment (AM): where the output format diverges from the expected answer.

CSE and CLE differ from IEE in that they assume syntactic or semantic code validity, but not necessarily accurate indexing.

As shown in Figure 3, computation errors are most frequent in text-based methods, reflecting LLMs' known weaknesses in arithmetic [26]. These models also struggle with large tables, where high information density increases both extraction errors and the risk of answer truncation due to context limits. The program-based paradigm alleviates these issues but still suffers from syntactic and logical flaws in code generation. In particular, HiTab's complex hierarchical indexing significantly raises the information extraction errors that occur in the program.

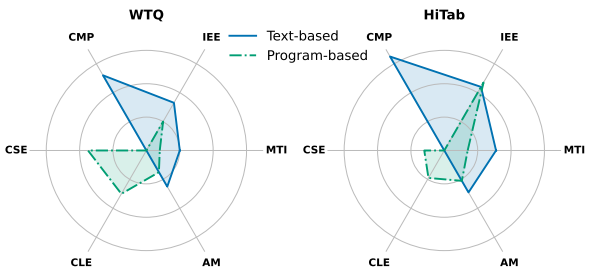

Figure 3. Error type distribution.

5.4. Ablation Study (RQ2, RQ3)

We conduct ablation studies (RQ2) on four datasets based on two base models to assess each component of *Table-r1*. As shown in Table 3, all modules enhance overall performance. Key findings include: (1) Cold start provides the largest gains, especially for weaker base models; (2) GRPO consistently boosts performance, with or without cold start; (3) Mix-GRPO yields notable improvements on structurally complex datasets (HiTab, AIT-QA).

 doi:10.20944/preprints202506.0552.v1

Table 3. Ablation results on four TR benchmarks with Qwen-7B and LLaMA-8B. Removing each module consistently degrades performance, verifying their complementary contributions.

	Module	WikiTQ	TabFact	HiTab	AiT-QA
Qwen-7B	*Table-r1 (w/ all modules)*	**76.5**	**85.3**	**76.7**	**89.9**
	Table-r1 (w/o all modules)	63.5 (↓13.0)	76.0 (↓9.3)	62.7 (↓14.0)	55.5 (↓34.4)
	- SSL	74.2 (↓2.3)	84.1 (↓1.2)	74.4 (↓2.3)	87.3 (↓2.6)
	- Cold Start	71.1 (↓5.4)	80.5 (↓4.8)	68.8 (↓7.9)	67.3 (↓22.6)
	- GRPO	71.7 (↓4.8)	81.2 (↓4.1)	70.4 (↓6.3)	83.3 (↓6.6)
	- Mix	75.1 (↓1.4)	84.4 (↓0.9)	73.6 (↓3.1)	86.4 (↓3.5)
LLaMA-8B	*Table-r1 (w/ all modules)*	**74.3**	**84.4**	**74.2**	**86.5**
	Table-r1 (w/o all modules)	55.3 (↓19.0)	65.2 (↓19.2)	49.2 (↓25.0)	48.8 (↓37.7)
	- SSL	72.2 (↓2.1)	83.0 (↓1.4)	70.9 (↓3.3)	83.4 (↓3.1)
	- Cold Start	67.5 (↓6.8)	77.3 (↓7.1)	63.8 (↓10.4)	63.5 (↓23.0)
	- GRPO	70.0 (↓4.3)	77.4 (↓7.0)	66.1 (↓8.1)	77.3 (↓9.2)
	- Mix	73.2 (↓1.1)	83.2 (↓1.2)	69.6 (↓4.6)	81.4 (↓5.1)

To examine mix-paradigm GRPO (RQ2), we analyze output template distributions and observe a strong preference for P-TR (WTQ: 95%, HiTab: 91%), supporting our hypothesis that P-TR better suits SLMs. Case studies further show that, with mix-paradigm enabled, the model selectively switches to text-based templates when contextually appropriate (The strategy is learned during training and tends to switch to T-TR when headers are dense and the question is a simple look-up. See Figure A7).

Table 4 further investigates the effectiveness of the original SSL task—LTI—and the necessity of incorporating continual learning. As shown, using only template-based annotation (w/o rewriting) leads to catastrophic forgetting. In contrast, introducing continual learning significantly improves model performance.

Table 4. Impact of LTI on Qwen-7B across four TR benchmarks. Label rewriting is crucial—without it, performance drops. Full LTI yields consistent gains across all datasets.

Module	WikiTQ	TabFact	HiTab	AiT-QA
Base Model	63.5	76.0	62.7	55.5
+ LTI (w/o rewriting)	59.2 (↓ 4.3)	71.4 (↓ 4.6)	58.2 (↓ 4.5)	50.1 (↓ 5.4)
+ LTI	65.8 (↑ 2.3)	77.5 (↑ 1.5)	65.7 (↑ 3.0)	59.1 (↑ 3.6)

To analyze the impact of RL and SFT on model performance (RQ3), we compare SC=5 and Pass@16 accuracy across five variants: Base, Origin (Base + SSL), Origin + RL, Distillation (Origin + Cold Start), and Distillation + RL. Results in Figure 4 yield three key observations: TabFact exhibits minimal change in potential due to its binary nature, where random sampling yields a high probability of correct answers. All components—SSL, distillation, and GRPO—improve both potential (Pass@16) and capability (SC=5), with SFT showing the most substantial gains. While distillation + GRPO achieves the highest capability, GRPO after distillation reduces potential. Moreover, GRPO still cannot fully convert potential into actual capability.

 doi:10.20944/preprints202506.0552.v1

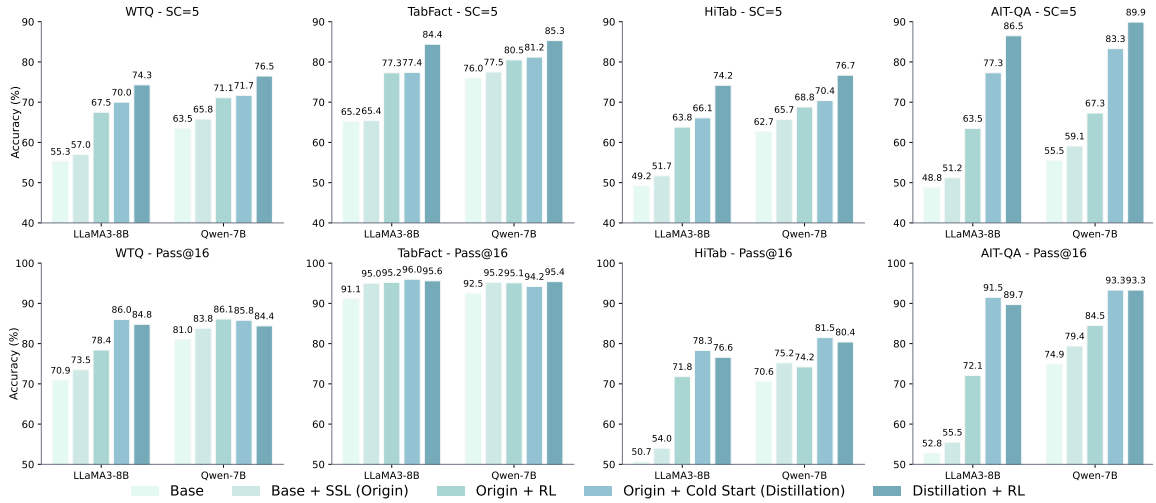

Figure 4. Comparison of SC=5 and Pass@16 accuracy across five model variants to assess the impact of SFT and RL. The results highlight the effects of SSL, distillation, and GRPO on model potential and capability.

5.5. Further Analysis (RQ4)

Prior work [40,41] has shown that GRPO is sensitive to training data difficulty. To further investigate this (RQ4), we conduct controlled experiments on WTQ, a benchmark with substantial P-TR challenges. Using execution feedback from a distilled Qwen-7B model, we categorize training examples into three levels: Easy: code compiles and yields correct answers (7.5k examples); Medium: code compiles but yields incorrect answers (3.5k); Hard: code fails to compile (2.3k). Each subset is trained separately for 2200 steps (20 instances per step), logging two reward signals: answer correctness (max 1.5) and compilation success (max 0.75). Test set accuracy is recorded every 100 steps.

Results (Figure 5) show that compilation reward degrades with increasing difficulty, while answer correctness drops sharply even at the medium level. Throughout the process, models trained on sorted subsets underperform those trained on randomly shuffled data (76.5% dev accuracy). We also tested training only on the Easy+Medium subsets, achieving a best result of 75.5%. These findings suggest that while random mixing yields the best overall performance, training on data closer to the model's current ability can approach optimal results when compute resources are limited.

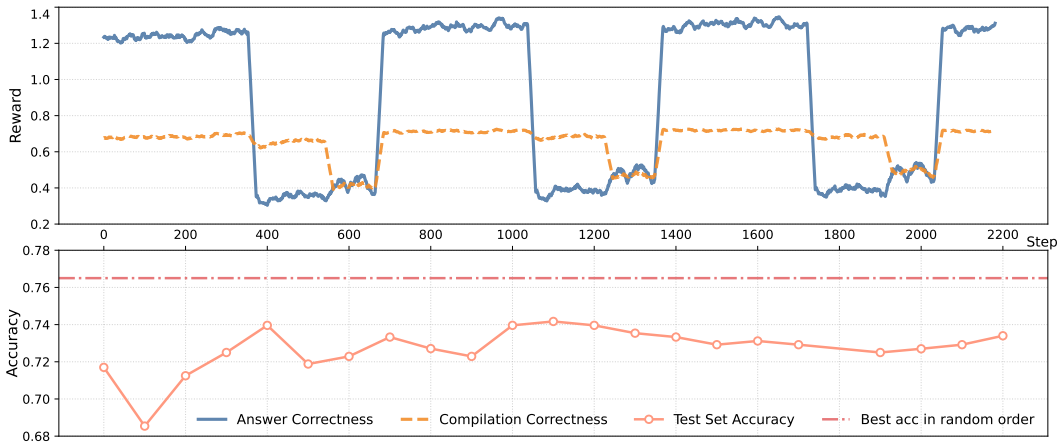

Figure 5. Training curves of Qwen-7B (after distillation) on the WTQ dataset sorted by difficulty levels (Easy → Medium → Hard) for 2200 steps. The plot shows reward signals during training and evaluation accuracy on the dev set. The red dashed line indicates the best accuracy achieved on WTQ with randomly shuffled training data.

6. Conclusion

We present *Table-r1*, a two-stage method that enhances SLM-based TR through an original SSL task and mix-paradigm GRPO. It effectively mitigates layout generalization and reasoning inconsistency

 doi:10.20944/preprints202506.0552.v1

issues in P-TR, achieving performance competitive with LLMs. Future work will explore broader RL strategies to advance TR capabilities further.

Appendix A. Limitations

1. Although our method has been evaluated on four popular public benchmarks—covering various types of semi-structured tables (e.g., WTQ, TabFact)—it may not generalize well to other types of tables, such as multimodal tables.

2. The SLMs used in this study refer to 7B or 8B-scale language models, and the proposed approach may not be directly applicable to smaller models such as 3B or 0.5B LMs.

3. Currently, the reward weighting for different functions in reinforcement learning still resembles neural network hyperparameters, heavily reliant on empirical tuning and lacking automation.

Appendix B. Distinction Between Table Reasoning and Text-to-SQL

While TR and Text-to-SQL (T2SQL) share the overarching goal of enabling LLMs to extract question-relevant information from tabular data, there remain fundamental differences between the two tasks—especially in light of recent work where symbolic approaches for TR also adopt SQL-style intermediate programs.

Data Assumptions and Table Regularity. T2SQL operates over well-structured relational databases, where tables conform to strict schema constraints: each column has a clearly defined type, keys and relationships are explicitly annotated, and the table format is normalized. In contrast, tables in table reasoning tasks—such as those found in enterprise spreadsheets or scraped HTML tables—are typically semi-structured: they may lack consistent formatting, contain noisy or nested entries, and often do not enforce column type constraints (e.g., a single column might contain both strings and floats).

Input Format and Model Access. Because T2SQL assumes clean schema-driven tables, models are typically given only the table schema (i.e., column headers and possibly data types or foreign key links), and must generate SQL queries accordingly. In Table Reasoning, however, the semi-structured and ad hoc nature of tables necessitates full-table access, requiring the model to parse and reason over raw cell content, layout artifacts, and potential multi-row/column dependencies.

Appendix C. Implementation Details

Our base models are Qwen2.5-Coder-7B and LLaMA3.1-8B-Instruct, with DeepSeek-v3 serving as the teacher model for the cold-start phase. For each dataset, we distill up to 5,000 samples. Both supervised fine-tuning (SFT) and reinforcement learning (RL) are implemented using LoRA, with a rank of 32 for SFT and 64 for RL. During RL, the decoding temperature is set to 0.85.

All experiments are conducted on NVIDIA RTX 4090 GPUs and NVIDIA H100 GPUs. The batch size is set to $4 \times 8 \times 8$, where 4 is the number of samples per batch, the first 8 corresponds to the number of responses generated per sample, and the second 8 is the number of gradient accumulation steps. We train for 1 epoch. We use the trl library for GRPO implementation and VLLM as the inference backend. The maximum token length is set to 2,400, and input tables are truncated to the first 10 rows. The model achieves its best performance at around 400 steps.

During the cold-start phase, the teacher model used is DeepSeek-v3, and the template employed for distillation is the P-TR template. The T-TR template was intentionally excluded because its inclusion was observed to degrade post-distillation performance.

 doi:10.20944/preprints202506.0552.v1

Appendix D. Predefined Operations for LTI and Reward Functions

Table A1. Pre-defined Operations for LTI

Operations	Notes
`row_swap()`	Swap the positions of two rows..
`column_swap()`	Swap the positions of two columns.
`row_deletion()`	Remove one or more rows.
`column_deletion()`	Remove one or more columns.
`extract_rows()`	Extract a subset of rows.
`extract_columns()`	Extract a subset of columns.
`extract_rows_having_cells()`	extract rows containing cells with specific values.
`extract_columns_having_cells()`	extract columns containing certain cell values.
`transpose()`	Transpose the table, i.e., the top header becomes the left header, and vice versa.

Table A2. Reward Functions

Reward Functions	Max Reward Value	Notes
`strict_format_reward()`	0.75	Verifies that the entire output is strictly wrapped by tags.
`ans_reward()`	1.5	Measures the similarity between the model output and the reference answer.
`comment_ratio`	0.45	Evaluates the proportion of comments in the generated Python code.
`multiple_python_blocks_penalty()`	1.0	Penalizes overly fragmented Python code with multiple separate blocks.
`compilation_correctness`	0.75	Checks whether the generated code compiles without errors.
`code_short_length_penalty()`	0.5	Penalizes code that is shorter than a defined length threshold.

Appendix E. Prompts

Appendix E.1. Prompts for T-TR

- **Prompts (Text-based)**

```
You are given a question and a preview of a table.
Each column header contains the column name.

Respond in the following format:
<reasoning>
...
</reasoning>
<answer>
Final Answer: [answer]
</answer>
```

Figure A1. Prompts for T-TR.

 doi:10.20944/preprints202506.0552.v1

Appendix E.2. Prompts for P-TR

- **Prompts (Program-based)**

```
You are given a question and a preview of a table stored in a pandas DataFrame called **df**.
Each column header contains both the column name and its data type (e.g., "Year\n(int64)"),
but you should only use the column name when writing code.

The Python code already includes the setup to read the table and define **df** — you do not need to
repeat that part.

Your task is to **complete the Python code** that answers the question based on the table contents.
In your reasoning, you may sketch a few candidate code approaches and briefly assess their correctness.
For example, whether your current column-handling code can be applied to each individual cell, etc.

Print the final answer using `print(variable)`, where `variable` holds the result.

Respond in the following format:
<reasoning>
...
</reasoning>
<code_solution>
```python
...
```
</code_solution>
"""
```

Figure A2. Prompts for P-TR.

 doi:10.20944/preprints202506.0552.v1

Appendix E.3. Prompts for the Mix-paradigm GRPO

- **Prompts (Mix)**

```
Each column header includes both the column name and its data type (e.g., "Year\n(int64)"),
but you should **only refer to the column name** in your code.
The setup code for reading the table and defining **df** is already provided — do not repeat it.
Your task is to answers the question based on the table.

Respond in one of the following formats:
<reasoning>
...
</reasoning>
<code_solution>
```python
...
```
</code_solution>
OR
<reasoning>
...
</reasoning>
<answer>
Final Answer: [Concise statement without question repetition]
</answer>

=== Response Protocol ===
1. REASONING PHASE:
<reasoning>
1. Task Analysis:
   - Identify required columns/rows from the question
   - Note any needed operations (filtering, aggregation, etc.)

2. Planning:
   - Draft pseudocode or step-by-step logic
   - Example: "First locate (...) then calculate (...) finally compare (...)"

3. Validation:
   - Verify assumptions about data structure
   - Cross-check with table preview
   - Reflect: "This approach works because (...)"

4. Decision:
   - [✓] Code solution needed / [✓] Direct answer possible
</reasoning>

2. EXECUTION PHASE:
[CHOOSE ONE]

=== OPTION A: Code-Based Solution ===
<code_solution>
python
# Comment each step
result = df.loc # Example
print(result) # output the final answer.
</code_solution>

=== OPTION B: Direct Answer ===
<answer>
Final Answer: [Concise statement without question repetition]
</answer>

=== Critical Reminders ===
1. Always use tuple indexing: df.loc[row_tuple, col_tuple]
2. Never assume single-level headers/indices
3. Validate against edge cases (missing hierarchies, etc.)"""
```

Figure A3. Prompts for the Mix-paradigm GRPO.

 doi:10.20944/preprints202506.0552.v1

Appendix F. Case Studies

- **Information Extraction Error (Program-based)**

Input:

Question: the mean depreciation rate for these selected machinery and equipment categories as a whole, weighted by their chained dollar investment shares, what was the percentage increased from 2002 to 2010?
Table caption: depreciation rates for select machinery and equipment categories
Table Preview:

	('1985 to 2001',)	('2002 to 2010',)	('1985 to 2010',)
	(float64)	(float64)	(float64)
('rate', 'asset description and code', '')			
('rate', 'asset description and code', 'office furniture, furnishing')	0.2	0.282	0.227
('rate', 'asset description and code', 'non-office furniture, furnishings and fixtures')	0.224	0.259	0.24
('rate', 'asset description and code', 'drilling and blasting equipment')	0.168	0.128	0.156
('rate', 'asset description and code', 'automobiles and major replacement parts')	0.295	0.287	0.279
('rate', 'asset description and code', 'trucks, vans, truck tractors, truck trailers and major replacement parts')	0.203	0.192	0.201
('rate', 'asset description and code', 'computer-assisted processing machinery and equipment')	0.142	0.134	0.134
('rate', 'asset description and code', 'safety and security equipment and other machinery and equipment')	0.145	0.185	0.167
('rate', 'mean, all assets', '')	0.195	0.256	0.218
('standard error', 'standard error of mean, all assets', '')	0.004	0.008	0.003

Code Solution:
```python
from utils import read_csv_
df = read_csv_('data/hitab/tables/hmt/1387.json', sep='\t')
```

Output:

<reasoning>
To find the percentage increase in the mean depreciation rate from 2002 to 2010 for the selected machinery and equipment categories, we need to:

1. Identify the mean depreciation rate for the period 2002 to 2010. This value is located in the row `('rate', 'mean, all assets', '')` and the column `('2002 to 2010',)`.
2. Identify the mean depreciation rate for the period 2002 to 2010. This value is located in the row `('rate', 'mean, all assets', '')` and the column `('1985 to 2010',)`.
3. Calculate the difference between the two rates.
4. Calculate the percentage increase by dividing the difference by the 1985 to 2010 rate and then multiplying by 100.

The percentage increase is given by the formula:
\[\text{Percentage Increase} = \left(\frac{\text{2002 to 2010 rate} - \text{1985 to 2010 rate}}{\text{1985 to 2010 rate}} \right) \times 100 \]

</reasoning>
<code_solution>
```python
# Access the mean depreciation rates for 2002-2010 and 1985-2010
rate_2002_2010 = df.loc[('rate', 'mean, all assets', ''), ('2002 to 2010',)]
rate_1985_2010 = df.loc[('rate', 'mean, all assets', ''), ('1985 to 2010',)]

# Calculate the percentage increase
percentage_increase = ((rate_2002_2010 - rate_1985_2010) / rate_1985_2010) * 100

print(percentage_increase)
```
</code_solution>

Output: 17.43119266055046
Golden Answer: 0.256

Figure A4. In P-TR, errors occur due to misunderstandings of the table's semantics. Specifically, the second-to-last row and second-to-last column of the table already provide the answer to the question, but the model fails to recognize this and instead incorrectly attempts to calculate the average from the existing numerical values.

 doi:10.20944/preprints202506.0552.v1

- **Information Extraction Error (Text-based)**

Input:

Question: how many drivers have no laps led?
Table caption: 2009 Honda 200
Table Preview:

	Pos	No.	Driver	Team	Laps	Time/Retired	Grid	Laps Led	Points
	(int64)	(int64)	(str)	(str)	(int64)	(str)	(int64)	(int64)	(int64)
0	1	9	Scott Dixon	Chip Ganassi Racing	85	1:46:05.7985	3	51	52
1	2	6	Ryan Briscoe	Penske Racing	85	29.7803	1	6	41
2	3	10	Dario Franchitti	Chip Ganassi Racing	85	30.0551	6	0	35
3	4	14	Ryan Hunter-Reay	A. J. Foyt Enterprises	85	33.7307	7	0	32
4	5	27	Hideki Mutoh	Andretti Green Racing	85	34.1839	11	0	30
5	6	26	Marco Andretti	Andretti Green Racing	85	46.7669	13	0	28
6	7	5	Paul Tracy	KV Racing Technology	85	49.702	10	0	26
7	8	2	Graham Rahal	Newman/Haas/Lanigan Racing	85	50.4517	4	0	24
8	9	2	Raphael Matos (R)	Luczo-Dragon Racing	85	51.2286	15	0	22
9	10	11	Tony Kanaan	Andretti Green Racing	85	52.081	8	0	20
10	11	6	Oriol Servià	Newman/Haas/Lanigan Racing	85	52.6215	14	0	19
11	12	3	Hélio Castroneves	Penske Racing	85	53.2362	5	0	18
12	13	18	Justin Wilson	Dale Coyne Racing	85	53.5768	2	28	17
13	14	33	Robert Doornbos (R)	HVM Racing	85	+ 1:10.0812	18	0	16
14	15	13	E. J. Viso	HVM Racing	84	+ 1 Lap	9	0	15
15	16	4	Dan Wheldon	Panther Racing	84	+ 1 Lap	17	0	14
16	17	20	Ed Carpenter	Vision Racing	84	+ 1 Lap	21	0	13
17	18	98	Richard Antinucci	Team 3G	83	+ 2 Laps	19	0	12
18	19	7	Danica Patrick	Andretti Green Racing	83	+ 2 Laps	12	0	12
19	20	24	Mike Conway (R)	Dreyer \& Reinbold Racing	69	Mechanical	16	0	12
20	21	23	Milka Duno	Dreyer \& Reinbold Racing	56	Handling	20	0	12

Output:

<reasoning>
To determine how many drivers have no laps led, we need to look at the "Laps Led" column in the table. All the drivers in this race completed 85 laps (as indicated in the "Laps" column), so we should check the "Laps Led" column for zeros.

Going through the "Laps Led" column, we see that all drivers have a value of 0 in this column.
</reasoning>
<answer>
Final Answer: All drivers (20 drivers) have no laps led.
</answer>

Output: 18
Golden Answer: 20

Figure A5. In T-TR, errors arise from information extraction errors of the table. Specifically, the second-to-last column contains non-zero cells, yet the model incorrectly assumes all cells to be zero. This also demonstrates that as table size increases, the model's ability to comprehend tabular information becomes progressively more challenging.

 doi:10.20944/preprints202506.0552.v1

- **Logical Errors in Code (Program-based)**

Input:

Question: where did he score his first international goal?
Table caption: Ismael Urzaiz
Table Preview:

	#	Date	Venue	Opponent	Score	Result	Competition
	(float64)	(str)	(str)	(str)	(str)	(str)	(str)
0	1	27 March 1999	Mestalla, Valencia, Spain	Austria	3-0	9-0	Euro 2000 qualifying
1	2	27 March 1999	Mestalla, Valencia, Spain	Austria	5-0	9-0	Euro 2000 qualifying
2	3	31 March 1999	Olimpico, Serravalle, San Marino	San Marino	0-3	0-6	Euro 2000 qualifying
3	4	8 September 1999	Vivero, Badajoz, Spain	Cyprus	1-0	8-0	Euro 2000 qualifying
4	5	8 September 1999	Vivero, Badajoz, Spain	Cyprus	2-0	8-0	Euro 2000 qualifying
5	6	8 September 1999	Vivero, Badajoz, Spain	Cyprus	4-0	8-0	Euro 2000 qualifying
6	7	26 January 2000	Cartagonova, Cartagena, Spain	Poland	2-0	3-0	Friendly
7	8	26 January 2000	Cartagonova, Cartagena, Spain	Poland	3-0	3-0	Friendly

Code Solution:
```python
from utils import read_csv_
df = read_csv_('data/original_data/wtq/csv/203-csv/766.tsv', sep='\t')
```

Output:

The task is to determine the venue where Ismael Urzaiz scored his first international goal. To achieve this, we need to find the earliest date in the "Date" column and then identify the corresponding "Venue".

Reasoning:
1. **Sorting the Data**: We need to sort the DataFrame by the "Date" column to ensure that the earliest date is at the top.
2. **Identifying the First Goal**: Once the data is sorted, the first row will contain the earliest date, and we can then extract the corresponding "Venue" from this row.

Code Solution:
```python
# Sort the DataFrame by the 'Date' column
df_sorted = df.sort_values(by='Date')

# Get the venue of the first game
first_venue = df_sorted.iloc[0]['Venue']

print(first_venue)
```

Output: Cartagonova, Cartagena, Spain
Golden Answer: Mestalla, Valencia, Spain

Figure A6. In P-TR, errors occur when the generated code solution lacks thorough consideration. Specifically, while the model is tasked with identifying the earliest timestamp, the implemented code merely performs a simple sort on the "Date" column without first converting the values into comparable numerical date formats.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 June 2025 doi:10.20944/preprints202506.0552.v1

- **Incorrect P-TR vs. correct T-TR on identical instance in HiTab**

Input:

Question: how many percentage points did donors between 35 and 54 years whose household income was in the highest quintile plan on claiming a tax credit?

Table caption: donors who plan on claiming a

Table Preview:

	('household income quintiles', 'total')	('household income quintiles', 'bottom two quintiles')	('household income quintiles', 'third quintile')	('household income quintiles', 'top two quintiles')
	(float64)	(float64)	(float64)	(float64)
('percentage', 'donation amount', '$100 to $249')	51.0	42.0	55.0	56.0
('percentage', 'donation amount', '$500 or more')	78.0	72.0	77.0	83.0
('percentage', 'sex', '')				
('percentage', 'age group', '15 to 34 years')	32.0	23.0	33.0	36.0
('percentage', 'age group', '35 to 54 years')	51.0	34.0	45.0	64.0
('percentage', 'age group', '55 years and older')	58.0	51.0	63.0	68.0
('percentage', 'region of residence', '')				
('percentage', 'region of residence', 'atlantic canada')	45.0	37.0	49.0	53.0
('percentage', 'region of residence', 'prairies')	58.0	52.0	59.0	63.0
('percentage', 'region of residence', 'alberta')	51.0	43.0	55.0	54.0

Code Solution:

```python
from utils import read_csv_
import pandas as pd
import numpy as np
df = read_csv_('data/hitab/tables/hmt/2551.json', sep='\t')```

**Program-based Output:**

<reasoning>
1. **Task Analysis**:
- Identify the relevant data: donors between 35 and 54 years old whose household income was in the highest quintile.
- Determine the percentage they plan to claim a tax credit.
- Calculate the percentage points difference from the total percentage.
2. **Planning**:
- Locate the row for age group '35 to 54 years' under the 'percentage' level.
- Find the column for 'top two quintiles' under the 'household income quintiles' level.
- Extract the percentage value from the DataFrame.
- Calculate the difference from the total percentage (which is 100%).
3. **Validation**:
- Verify that the row and column tuples are correct based on the table preview.
- Ensure the DataFrame structure and indexing are as expected.
4. **Decision**: - Code solution needed.
</reasoning>
<code_solution>
```python
# Extract the percentage for donors between 35 and 54 years whose household income was in the highest quintile
percentage = df.loc[('percentage', 'age group', '35 to 54 years'), ('household income quintiles', 'top two quintiles')]
# Calculate the percentage points difference from the total percentage (100%)
percentage_points_difference = 100 - percentage
print(percentage_points_difference)
```
</code_solution>
Execution Results: 36.0
Golden Answer: 64.0

Text-based Output:

<reasoning>
1. **Task Analysis**:
- Identify the relevant row: ('percentage', 'age group', '35 to 54 years')
- Identify the relevant column: ('household income quintiles', 'top two quintiles')
- Extract the value from the DataFrame at this location.

2. **Planning**:
- Use tuple indexing to locate the specified row and column.
- Extract the value from the DataFrame.
- Since the question asks for the percentage points, we already have the required percentage directly from the DataFrame.

3. **Validation**:
- Verify that the DataFrame has the expected structure with hierarchical indices.
- Cross-check the table preview to ensure the row and column exist.
- Reflect: "This approach works because the DataFrame structure matches the expected hierarchical indices and the required data is directly accessible."

4. **Decision**:
- Direct answer possible
</reasoning>

<answer>
Final Answer: 64.0
</answer>

Golden Answer: 64.0

Figure A7. An example where the P-TR method fails but the T-TR method succeeds, due to the table layout being too complex for the program to correctly use headers as indices.
```

 doi:10.20944/preprints202506.0552.v1

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See Abstract and Section 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Appendix A.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

 doi:10.20944/preprints202506.0552.v1

Answer: [NA]
Justification: The paper does not include theoretical results.
Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**
Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?
Answer: [Yes]
Justification: See Section 5.1 and Appendix C.
Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

 doi:10.20944/preprints202506.0552.v1

5. **Open access to data and code**
   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?
   Answer: [Yes]
   Justification: We provide code along with sufficient instructions in supplemental material.
   Guidelines:
   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**
   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?
   Answer: [Yes]
   Justification: See Section 5.1, Appendix C and supplemental material.
   Guidelines:
   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**
   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?
   Answer: [Yes]
   Justification: Standard deviation in Table 2.
   Guidelines:
   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

 doi:10.20944/preprints202506.0552.v1

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

 doi:10.20944/preprints202506.0552.v1

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: See Section 5.1.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

 doi:10.20944/preprints202506.0552.v1

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**
    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?
    Answer: [NA]
    Justification: The paper does not release new assets.
    Guidelines:
    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**
    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?
    Answer: [NA]
    Justification: The paper does not involve crowdsourcing nor research with human subjects.
    Guidelines:
    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**
    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?
    Answer: [NA]
    Justification: The paper does not involve crowdsourcing nor research with human subjects.
    Guidelines:
    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

 doi:10.20944/preprints202506.0552.v1

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**
    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.
    Answer: [Yes]
    Justification: LLMs for reasoning and distillation.
    Guidelines:
    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

References

1. Zhang, X.; Wang, D.; Dou, L.; Zhu, Q.; Che, W. A Survey of Table Reasoning with Large Language Models. *arXiv preprint arXiv:2402.08259* **2024**.
2. Jin, R.; Li, Y.; Qi, G.; Hu, N.; Li, Y.F.; Chen, J.; Wang, J.; Chen, Y.; Min, D.; Bi, S. HeGTa: Leveraging Heterogeneous Graph-enhanced Large Language Models for Few-shot Complex Table Understanding. In Proceedings of the Proceedings of the AAAI Conference on Artificial Intelligence, 2025, Vol. 39, pp. 24294–24302.
3. Jin, R.; Wang, J.; Tan, W.; Chen, Y.; Qi, G.; Hao, W. TabPrompt: Graph-based Pre-training and Prompting for Few-shot Table Understanding. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2023.
4. Lu, W.; Zhang, J.; Zhang, J.; Chen, Y. Large language model for table processing: A survey. *arXiv preprint arXiv:2402.05121* **2024**.
5. Chen, W. Large Language Models are few(1)-shot Table Reasoners. *ArXiv* **2022**, *abs/2210.06710*.
6. Cheng, Z.; Xie, T.; Shi, P.; Li, C.; Nadkarni, R.; Hu, Y.; Xiong, C.; Radev, D.R.; Ostendorf, M.; Zettlemoyer, L.; et al. Binding Language Models in Symbolic Languages. *ArXiv* **2022**, *abs/2210.02875*.
7. Deng, N.; Mihalcea, R. Rethinking Table Instruction Tuning. *arXiv preprint arXiv:2501.14693* **2025**.
8. Ji, D.; Zhu, L.; Gao, S.; Xu, P.; Lu, H.; Ye, J.; Zhao, F. Tree-of-Table: Unleashing the Power of LLMs for Enhanced Large-Scale Table Understanding. *arXiv preprint arXiv:2411.08516* **2024**.
9. Li, Q.; Huang, C.; Li, S.; Xiang, Y.; Xiong, D.; Lei, W. GraphOTTER: Evolving LLM-based Graph Reasoning for Complex Table Question Answering. In Proceedings of the Proceedings of the 31st International Conference on Computational Linguistics; Rambow, O.; Wanner, L.; Apidianaki, M.; Al-Khalifa, H.; Eugenio, B.D.; Schockaert, S., Eds., Abu Dhabi, UAE, 2025; pp. 5486–5506.
10. Srivastava, G.; Cao, S.; Wang, X. Towards reasoning ability of small language models. *arXiv preprint arXiv:2502.11569* **2025**.
11. Zhang, T.; Yue, X.; Li, Y.; Sun, H. TableLlama: Towards Open Large Generalist Models for Tables. *ArXiv* **2023**, *abs/2311.09206*.
12. Lu, X.; Pan, L.; Ma, Y.; Nakov, P.; Kan, M.Y. TART: An Open-Source Tool-Augmented Framework for Explainable Table-based Reasoning. *arXiv preprint arXiv:2409.11724* **2024**.
13. Su, A.; Wang, A.; Ye, C.; Zhou, C.; Zhang, G.; Chen, G.; Zhu, G.; Wang, H.; Xu, H.; Chen, H.; et al. Tablegpt2: A large multimodal model with tabular data integration. *arXiv preprint arXiv:2411.02059* **2024**.
14. Chen, W.; Ma, X.; Wang, X.; Cohen, W.W. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588* **2022**.
15. Liu, T.; Wang, F.; Chen, M. Rethinking Tabular Data Understanding with Large Language Models. *arXiv preprint arXiv:2312.16702* **2023**.

 doi:10.20944/preprints202506.0552.v1

16. Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Bi, X.; Zhang, H.; Zhang, M.; Li, Y.; Wu, Y.; et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300* **2024**.
17. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q.V.; Zhou, D.; et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* **2022**, *35*, 24824–24837.
18. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. *ArXiv* **2020**, *abs/2005.14165*.
19. Zhao, B.; Ji, C.; Zhang, Y.; He, W.; Wang, Y.; Wang, Q.; Feng, R.; Zhang, X. Large language models are complex table parsers. *arXiv preprint arXiv:2312.11521* **2023**.
20. Xing, J.; He, Y.; Zhou, M.; Dong, H.; Han, S.; Zhang, D.; Chaudhuri, S. Table-LLM-Specialist: Language Model Specialists for Tables using Iterative Generator-Validator Fine-tuning. *arXiv preprint arXiv:2410.12164* **2024**.
21. Sui, Y.; Zhou, M.; Zhou, M.; Han, S.; Zhang, D. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In Proceedings of the Proceedings of the 17th ACM International Conference on Web Search and Data Mining, 2024, pp. 645–654.
22. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. React: Synergizing reasoning and acting in language models. In Proceedings of the International Conference on Learning Representations (ICLR), 2023.
23. Li, P.; He, Y.; Yashar, D.; Cui, W.; Ge, S.; Zhang, H.; Fainman, D.R.; Zhang, D.; Chaudhuri, S. Table-gpt: Table-tuned gpt for diverse table tasks. *arXiv preprint arXiv:2310.09263* **2023**.
24. Zhuang, A.; Zhang, G.; Zheng, T.; Du, X.; Wang, J.; Ren, W.; Huang, S.W.; Fu, J.; Yue, X.; Chen, W. Structlm: Towards building generalist models for structured knowledge grounding. *arXiv preprint arXiv:2402.16671* **2024**.
25. He, X.; Liu, Y.; Zhou, M.; He, Y.; Dong, H.; Han, S.; Yuan, Z.; Zhang, D. TableLoRA: Low-rank Adaptation on Table Structure Understanding for Large Language Models. *arXiv preprint arXiv:2503.04396* **2025**.
26. Gao, L.; Madaan, A.; Zhou, S.; Alon, U.; Liu, P.; Yang, Y.; Callan, J.; Neubig, G. Pal: Program-aided language models. In Proceedings of the International Conference on Machine Learning. PMLR, 2023, pp. 10764–10799.
27. Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Hambro, E.; Zettlemoyer, L.; Cancedda, N.; Scialom, T. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems* **2024**, *36*.
28. Ahn, J.; Verma, R.; Lou, R.; Liu, D.; Zhang, R.; Yin, W. Large Language Models for Mathematical Reasoning: Progresses and Challenges. In Proceedings of the Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop; Falk, N.; Papi, S.; Zhang, M., Eds., St. Julian's, Malta, 2024; pp. 225–237.
29. Zhang, Y.; Henkel, J.; Floratou, A.; Cahoon, J.; Deep, S.; Patel, J.M. ReAcTable: Enhancing ReAct for Table Question Answering. *ArXiv* **2023**, *abs/2310.00815*.
30. Wang, Z.; Zhang, H.; Li, C.L.; Eisenschlos, J.M.; Perot, V.; Wang, Z.; Miculicich, L.; Fujii, Y.; Shang, J.; Lee, C.Y.; et al. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *arXiv preprint arXiv:2401.04398* **2024**.
31. Ye, Y.; Hui, B.; Yang, M.; Li, B.; Huang, F.; Li, Y. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In Proceedings of the Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2023, pp. 174–184.
32. Zhang, H.; Ma, Y.; Yang, H. ALTER: Augmentation for Large-Table-Based Reasoning. *arXiv preprint arXiv:2407.03061* **2024**.
33. Wang, Z.; Fried, D.; Neubig, G. Trove: Inducing verifiable and efficient toolboxes for solving programmatic tasks. *arXiv preprint arXiv:2401.12869* **2024**.
34. Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems* **2024**, *36*.
35. Yang, Z.; Du, Z.; Zhang, M.; Du, W.; Chen, J.; Duan, Z.; Zhao, S. Triples as the Key: Structuring Makes Decomposition and Verification Easier in LLM-based TableQA. In Proceedings of the The Thirteenth International Conference on Learning Representations, 2025.
36. Zhao, Y.; Yin, H.; Zeng, B.; Wang, H.; Shi, T.; Lyu, C.; Wang, L.; Luo, W.; Zhang, K. Marco-o1: Towards open reasoning models for open-ended solutions. *arXiv preprint arXiv:2411.14405* **2024**.

 doi:10.20944/preprints202506.0552.v1

37. Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* **2025**.
38. Team, K.; Du, A.; Gao, B.; Xing, B.; Jiang, C.; Chen, C.; Li, C.; Xiao, C.; Du, C.; Liao, C.; et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599* **2025**.
39. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* **2017**.
40. Liu, Z.; Chen, C.; Li, W.; Qi, P.; Pang, T.; Du, C.; Lee, W.S.; Lin, M. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783* **2025**.
41. Zeng, W.; Huang, Y.; Liu, Q.; Liu, W.; He, K.; Ma, Z.; He, J. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892* **2025**.
42. Yue, Y.; Chen, Z.; Lu, R.; Zhao, A.; Wang, Z.; Song, S.; Huang, G. Does Reinforcement Learning Really Incentivize Reasoning Capacity in LLMs Beyond the Base Model? *arXiv preprint arXiv:2504.13837* **2025**.
43. Cheng, Z.; Dong, H.; Wang, Z.; Jia, R.; Guo, J.; Gao, Y.; Han, S.; Lou, J.G.; Zhang, D. HiTab: A Hierarchical Table Dataset for Question Answering and Natural Language Generation. *ArXiv* **2021**, *abs/2108.06712*.
44. Katsis, Y.; Chemmengath, S.; Kumar, V.; Bharadwaj, S.; Canim, M.; Glass, M.; Gliozzo, A.; Pan, F.; Sen, J.; Sankaranarayanan, K.; et al. AIT-QA: Question Answering Dataset over Complex Tables in the Airline Industry. *NAACL-HLT 2022* **2022**, p. 305.
45. Wu, T.; Luo, L.; Li, Y.F.; Pan, S.; Vu, T.T.; Haffari, G. Continual learning for large language models: A survey. *arXiv preprint arXiv:2402.01364* **2024**.
46. Yang, Z.; Pang, T.; Feng, H.; Wang, H.; Chen, W.; Zhu, M.; Liu, Q. Self-distillation bridges distribution gap in language model fine-tuning. *arXiv preprint arXiv:2402.13669* **2024**.
47. Nye, M.; Andreassen, A.J.; Gur-Ari, G.; Michalewski, H.; Austin, J.; Bieber, D.; Dohan, D.; Lewkowycz, A.; Bosma, M.; Luan, D.; et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114* **2021**.
48. DeepSeek-AI.; Liu, A.; Feng, B.; Xue, B.; Wang, B.; Wu, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; et al. DeepSeek-V3 Technical Report, 2024, [arXiv:cs.CL/2412.19437].
49. Pasupat, P.; Liang, P. Compositional Semantic Parsing on Semi-Structured Tables. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, 2015.
50. Chen, W.; Wang, H.; Chen, J.; Zhang, Y.; Wang, H.; Li, S.; Zhou, X.; Wang, W.Y. TabFact: A Large-scale Dataset for Table-based Fact Verification. In Proceedings of the International Conference on Learning Representations, 2020.
51. Jaech, A.; Kalai, A.; Lerer, A.; Richardson, A.; El-Kishky, A.; Low, A.; Helyar, A.; Madry, A.; Beutel, A.; Carney, A.; et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720* **2024**.
52. Yang, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Li, C.; Liu, D.; Huang, F.; Wei, H.; et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115* **2024**.
53. Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Vaughan, A.; et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* **2024**.
54. Zhao, B.; Ji, C.; Zhang, Y.; He, W.; Wang, Y.; Wang, Q.; Feng, R.; Zhang, X. Large language models are complex table parsers. *arXiv preprint arXiv:2312.11521* **2023**.
55. Yu, P.; Chen, G.; Wang, J. Table-critic: A multi-agent framework for collaborative criticism and refinement in table reasoning. *arXiv preprint arXiv:2502.11799* **2025**.
56. Mao, Q.; Liu, Q.; Li, Z.; Cheng, M.; Zhang, Z.; Li, R. PoTable: Programming Standardly on Table-based Reasoning Like a Human Analyst. *arXiv preprint arXiv:2412.04272* **2024**.
57. Cao, L. TableMaster: A Recipe to Advance Table Understanding with Language Models. *arXiv preprint arXiv:2501.19378* **2025**.
58. Zhang, Z.; Gao, Y.; Lou, J.G. e5: Zero-shot hierarchical table analysis using augmented LLMs via explain, extract, execute, exhibit and extrapolate. In Proceedings of the Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), 2024, pp. 1244–1258.