TAREX: Reinforcement Learning for Code-Driven Table Reasoning

Fangyu Lei[1,2*], **Jinxiang Meng**[1,2*], **Yiming Huang**[3], **Shizhu He**[1,2], **Jun Zhao**[1,2], **Kang Liu**[1,2†]

[1]The Key Laboratory of Cognition and Decision Intelligence for Complex Systems,
Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]Independent Researcher
leifangyu2022@ia.ac.cn, kliu@nlpr.ia.ac.cn

Abstract

Automatically solving table reasoning tasks remains challenging due to three main factors: (1) diverse and hierarchical table structures that hinder comprehension, (2) the heavy reliance on complex logical and numerical reasoning—which makes purely text-based methods prone to hallucinations—and (3) the necessity of multi-step processing to handle intricate tasks involving multiple and lengthy tables. To address these challenges, we introduce TAREX, a novel framework that unifies table representation, integrates code-driven execution, and supports interactive multi-step reasoning. TAREX employs a reinforcement learning-based training pipeline to optimize its reasoning policy for complex tasks. Experimental results show that TAREX achieves state-of-the-art performance across a wide range of table reasoning benchmarks, both in-domain and out-of-domain. These include fundamental tasks such as table question answering (TQA) and table fact verification (TFV), as well as advanced tabular data analysis tasks. The results highlight TAREX's effectiveness and scalability in advancing automated table reasoning.

Introduction

Tabular data provide a foundational structure for organizing, storing, and presenting information—the cornerstone of data analysis, data engineering, and countless real-world applications (Borisov et al. 2022; Cao et al. 2024). Table reasoning—extracting insights, performing logical inferences, and generating precise responses from structured data—has become a critical task. It spans diverse operations such as table question answering (Zhong, Xiong, and Socher 2017; Pasupat and Liang 2015), fact verification (Chen et al. 2019), and text-to-SQL generation (Li et al. 2024; Lei et al. 2024), while supporting a variety of table formats. The emergence of Large Language Models (LLMs) (OpenAI 2023; Anthropic 2024) has markedly advanced these capabilities, and state-of-the-art frameworks now increasingly rely on such models.

Prompt engineering tailors task-specific pipelines for table reasoning (Cheng et al. 2022b; Wang et al. 2024), yet its scalability and cross-task generalizability remain hampered by extensive manual tuning. Supervised fine-tuning (SFT) sharpens an LLM's grasp of tabular structures (Zhang et al. 2024a; Zha et al. 2023; Zhang et al. 2024b), but imitation-learning biases curb its adaptability (Chu et al. 2025; Guo et al. 2025), leaving it underpowered on lengthy tables and multi-step reasoning challenges.

Recent advanced LLMs demonstrate that reinforcement learning with verifiable rewards (RLVR) has driven significant performance improvements in domains such as mathematical reasoning (Yu et al. 2025; Zhang, song Tang, and Hao 2025) and code generation (Luo et al. 2025a). RL holds the potential to acquire high-quality reasoning paths and can explore solution patterns more deeply. We explore applying RLVR to table reasoning, aiming to enhance LLMs' capability in handling complex tables and sophisticated tasks.

Table reasoning tasks face three main challenges: 1) The diverse hierarchical structures of tables make structural understanding challenging for models. 2) Such tasks require extensive logical and numerical reasoning, and purely text-based approaches are highly prone to hallucinations. 3) Some complex table reasoning tasks cannot be solved in a single step and require multiple steps; meanwhile, some tables are often lengthy, necessitating models to process long contexts.

In this paper, we introduce TAREX, a unified code-execution RL method specifically designed for table reasoning. *(1) Unified executable table*: We address the challenge of diverse table structure by unifying the model's observation space, consolidating various table types into a carefully designed, standardized `pandas dataframe` format. This format facilitates programmatic access and manipulation of the table data using code. *(2) Code Execution RL*: TAREX employs an interactive approach combining code execution and text-based actions to form a hybrid action space. By generating and executing code for precise calculations and data transformations, the model interacts with tables more accurately, effectively mitigating hallucinations common in text-only methods. This process empowers models to explore tables freely, perform intricate reasoning, and generate high-quality code/text actions with reasoned traces. *(3) Interactive multi-step reasoning process*: Furthermore, the inherent multi-step code interaction naturally enables decomposing and executing table operations, obviating the need for full-table content processing in a single step. This capability directly addresses challenges from long tables and complex tasks.

Our contributions are as follows:

*Equal contribution.
†Corresponding author.

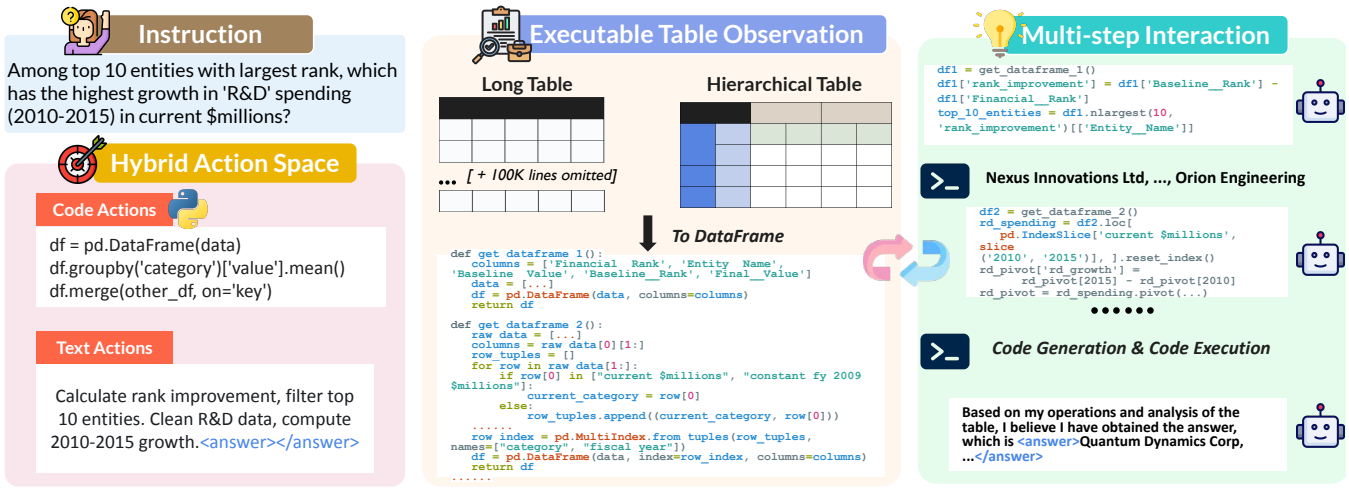

Figure 1: TAREX is a RL-based unified table reasoning framework. It tackles core challenges (structural diversity, reasoning hallucinations, complex task handling) via key mechanisms: unified executable table representation (*Executable Table Observation*), code-driven execution (*Hybrid Action Space*), and interactive multi-step reasoning (*Multi-step Interaction*).

- We introduce TAREX, a unified code-execution-based reinforcement learning framework for table reasoning, integrating three core technical components: a unified executable table representation to address diverse table structures, code-execution RL with a hybrid action space to mitigate hallucinations in logical and numerical reasoning, and an interactive multi-step reasoning process to handle complex tasks and long tables.
- We pioneer the application of multi-turn tool-integrated reinforcement learning to table reasoning tasks, supported by rigorous data preprocessing, task-specific reward design, and tailored training strategies, laying a robust technical foundation for RL-driven table reasoning.
- Through extensive experiments, TAREX delivers state-of-the-art performance across most table reasoning benchmarks, validating its effectiveness and adaptability across diverse table reasoning scenarios and laying groundwork for future advancements in this field.

TAREX

Problem Formulation

Given a natural language query Q and tables $\mathcal{T} = \{T_1, \ldots, T_n\}$ ($n \geq 1$), the goal is to derive an answer A. TAREX operates via an iterative process with a hybrid action space: each action a_t is either a text-only thought or a thought paired with executable Python code (for table operations such as filtering, aggregation, or joining). Code execution yields an observation O (e.g., subtable, transformed table, numerical result), and iteration terminates when $a_t =$ "final answer". A trajectory over T iterations is formalized as:

$$H = (\tau_0, a_0, o_0, \ldots, \tau_i, a_i, o_i, \ldots, \tau_T, a_T) \quad (1)$$

where τ_i, a_i, o_i denote the thought, action, and observation at iteration i; $a_T =$ "final answer" triggers the derivation of answer A. The hybrid nature of a_i is emphasized: a_i takes either a text-only form (pure thought) or a combination of thought and Python code. At step t, τ_t and a_t are sampled from a policy conditioned on prior context: $\pi(a_t, \tau_t | H_{t-1})$.

Unified Table Interaction Framework

Observation space. Current table representation poses three key challenges for reasoning: (1) Text-based table representations (e.g., markdown) hinder conversion to code-executable formats, limiting interaction between code and tables. (2) Structural diversity—including irregular layouts, nested headers, and hierarchical organization—complicates unified formatting; further, ordinary flattened tables lose important structural information. (3) Lengthy tables occupy significant amounts of LLM context, indicating the need for dynamic observation mechanisms: LLMs need not process the full table at once but only relevant sub-tables.

Thus, we propose using `DataFrame` (from `pandas`) due to three key advantages: (1) it supports direct execution via Python code; (2) its unified indexed structure enables accurate representation of nested and hierarchical tables; (3) it facilitates efficient observation of very long tables.

Converting any table into a `DataFrame` follows a structured process: first, extract its raw structure (including row/column coordinates, cell content, merging relationships) from diverse sources. Next, parse hierarchical elements such as nested headers and merged cells to clarify their logical relationships. This structured information is then mapped to a `DataFrame`, with `MultiIndex` employed to handle hierarchical headers when necessary. Additionally, the framework supports dynamic access to subsets of long tables, enabling efficient processing without loading the entire table at once.

Action space. At each interaction step, the agent employs a hybrid action space grounded in `DataFrame`-based observations, consisting of an explicit thought (τ_t) and a concrete action (a_t). τ_t performs adaptive reasoning over task goals, prior context, and current `DataFrame` states to guide strategy refine-

ment. a_t operationalizes this reasoning—either as text-only (to deepen logical analysis) or as a pairing of thought with executable Python code, with `</python>` marking the end of the code block. For any table, a predefined `get_dataframe()` function is available within the code block for invoking the target table data; critically, the observation O_t is derived not from the function itself, but from the output generated when the code explicitly prints the `DataFrame`.

The `pandas`-based code translates high-level intentions into precise tabular operations (e.g., filtering, aggregation, joins), yielding an observation (O_t: a subtable, numerical result, or updated `DataFrame` state) that feeds into the next iteration. This design balances flexible textual reasoning with precise programmatic table manipulation, enabling robust handling of complex multi-step table-based reasoning tasks.

Data Collection

Data sourcing. We curate training data from public datasets (Table 1) emphasizing two key diversity dimensions: First, diverse **table format types**, spanning three categories: (1) *Simple Column-Header Tables* (basic CSV/HTML-like with single column headers); (2) *Spreadsheet Tables* (hierarchical/multi-level headers, dual row-column headers, mixed data types like text, numbers, formulas); (3) *Embedded/Semi-Structured Tables* (within documents such as PDFs, Word files, webpages, with surrounding or interleaved text). Second, diverse **task types**, encompassing two common types: Table question answering and table fact verification. We then follow the unified observation space approach to convert these datasets uniformly into `Dataframe` format.

Difficulty filtering. To enhance RL data quality and focus on challenging, learnable samples, inspired by Cheng et al. (2025), we use a *difficulty-driven filtering mechanism* with two goals: (i) prioritize examples with sufficient reasoning complexity; (ii) rigorously exclude noisy or unstable samples. This mechanism calculates success probabilities of a weak model (M_{weak}, `Qwen2.5-7B-Instruct`) and a strong model (M_{strong}, `Qwen3-30B-A8B`) across $N = 16$ runs per sample, denoted P_{weak} and P_{strong}. Samples are discarded if they meet any of the following: *(1) Overly simple*: $P_{\text{weak}} \geq \frac{15}{16}$. These are reliably solved by the weak model, offering little value for RL improvement. *(2) Potentially noisy*: $P_{\text{strong}} = 0$. Consistently failed by the strong model, often due to ambiguous tasks, mislabels, or incompatibility with RL (e.g., needing extreme precision or external tools). *(3) Anomalous*: $P_{\text{weak}} > P_{\text{strong}}$. Occurs when the weak model reliably matches labels, typically from incorrect ground truth (likely due to the weak model memorizing flawed pairs during training), risking misleading RL by reinforcing errors.

RL Training Paradigm

Algorithm. Specifically, we employ GRPO's advantage estimation method—characterized by group-internal computation with normalization within sample groups (Shao et al. 2024)—while incorporating key components from DAPO (Yu et al. 2025), particularly the *clip-higher* trick. For the *clip-higher* trick, we set the low and high clip ratios to 0.2 and 0.28, respectively. This trick ensures that the ratio of the current policy's probability to the old policy's probability is clipped within a specific range, preventing overly large updates and stabilizing training. Notably, we exclude both entropy loss and KL divergence loss in our training objective. The corresponding formulation is presented as follows:

$$\begin{aligned}\mathcal{J}(\theta) = \mathbb{E}_{(Q,a_T)\sim\mathcal{D},\{H_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|Q)} \\ \left[\frac{1}{\sum_{i=1}^{G}|H_i|}\sum_{i=1}^{G}\sum_{t=1}^{|H_i|}\min\Big(r_{i,t}(\theta)\hat{A}_{i,t},\right. \\ \left.\text{clip}\left(r_{i,t}(\theta), 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}}\right)\hat{A}_{i,t}\Big)\right] \\ \text{s.t.} \quad 0 < |\{H_i \mid \text{is\_equivalent}(a_T, H_i)\}| < G,\end{aligned} \tag{2}$$

$$r_{i,t}(\theta) = \frac{\pi_\theta(a_{i,t},\tau_{i,t}|Q,H_{i,<t})}{\pi_{\theta_{\text{old}}}(a_{i,t},\tau_{i,t}|Q,H_{i,<t})}, \hat{A}_{i,t} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^{G})}{\text{std}(\{R_i\}_{i=1}^{G})} \tag{3}$$

In multi-turn agent scenarios, trajectories terminate either on timeout (due to prolonged generation or environment execution) or upon reaching the maximum number of environment steps. We use **compact filtering** to exclude such trajectories—those hitting maximum context, step limits, or timeouts. This mitigates or delays reward collapse during training, curbs excessive per-step deliberation, and encourages long-form reasoning across steps. We mask out the feedback output within the `<python></python>` blocks from loss computation.

Reward design. To ensure robust evaluation, we adapt a rule-based reward combining format and outcome validation:

$$R_{\text{total}} = 0.1 \times R_{\text{fmt}} + 0.9 \times R_{\text{ans}}$$

The outcome reward ($R_{\text{ans}} \in \{0, 1\}$) assesses answer correctness via two metrics: exact match and F1 thresholding (for long strings), **Exact Match (EM)**, which is 1 if the predicted answer ($\hat{y}$) exactly matches the ground-truth (y) and 0 otherwise (EM = 1 if $\hat{y} = y$, else 0); and **F1 Score**, computed from the precision and recall of tokens in the predicted answer (A) vs. the reference (A_{ref}) — where precision = $\frac{|\text{Tokens}(A)\cap\text{Tokens}(A_{\text{ref}})|}{|\text{Tokens}(A)|}$, recall = $\frac{|\text{Tokens}(A)\cap\text{Tokens}(A_{\text{ref}})|}{|\text{Tokens}(A_{\text{ref}})|}$, and F1 = $2 \cdot \frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$ — with tokens obtained by splitting into words/subwords and ignoring case and punctuation (unless specified otherwise):

$$R_{\text{ans}} = \begin{cases} 1 & \text{if } A = A_{\text{ref}} \text{ (short answers)} \\ 1 & \text{if F1}(A, A_{\text{ref}}) \geq \phi \text{ (long strings)} \\ 0 & \text{otherwise} \end{cases}$$

Dataset	Table Type	Task Type	#Origin	#After Filtering	Metric
WikiTQ	Simple	TQA	14,152	9,472	EM
MultiHiertt	Semi-Structured	TQA	7,830	5,119	F1
FinQA	Semi-Structured	TQA	6,251	4,333	F1
TAT-QA	Semi-Structured	TQA	13,210	8,085	F1
HiTab	Spreadsheet	TQA	7,417	4,888	EM
TabFact	Simple	TFV	24,101	15,087	EM

Table 1: Statistics of TAREX's training datasets, where the symbol # indicates the number of samples.

The format reward (R_{fmt}) enforces structural validity by verifying proper use of tags (</think>, <python>, </think>) in outputs. The weight allocation prioritizes answer accuracy while ensuring format compliance.

Strategic trajectory filtering. We filter out problematic trajectories during training. By analyzing training logs from both multi-turn and single-turn sessions, we identify a key source of instability in end-to-end reinforcement learning (RL): a phenomenon we term "stagnant turns" (Xue et al. 2025). Stagnant turns refer to unproductive steps that neither generate executable tool calls (i.e., complete code blocks) nor provide final answers. Such turns typically contain fragmented code or repetitive sentences, often triggered by the premature generation of end-of-sequence (EOS) tokens.

Experiment

Setup

Benchmarks. For in-domain evaluation, we select diverse benchmarks including FinQA (Chen et al. 2021b), HiTab (Cheng et al. 2022a), MultiHiertt (Zhao et al. 2022), TabFact (Chen et al. 2019), TAT-QA (Zhu et al. 2021), Feverous (Aly et al. 2021), and WikiTQ (Pasupat and Liang 2015). For out-of-domain assessment—designed to test generalization and robustness—we employ more challenging datasets: AIT-QA (Katsis et al. 2022) and TableBench (Wu et al. 2025b) (excluding the visualization task) are out-of-domain table reasoning datasets; OTT-QA (Chen et al. 2021a) is a large-scale open-domain question answering dataset that requires multi-hop inference across both retrieved tabular data and unstructured text from Wikipedia; TabularGSM (Tian et al. 2025) is a benchmark for table-based reasoning tasks spanning table complexities and trap problems, and we use its *pure test* setting; MultiModalQA (Talmor et al. 2021) is a dataset requiring joint reasoning over text, tables, and images for question answering, from which we retain only the parts that do not require images; RealHiTBench (Wu et al. 2025a) is a comprehensive benchmark for evaluating LLMs on complex hierarchical tables.

Baselines. We compare our method against large open-source LLMs, proprietary LLMs, and the table-specific domain model TableGPT2 (Su et al. 2024). We also compare TAREX with recent RL-based methods, Table-R1 (Yang et al. 2025) and Reasoning-Table (Lei et al. 2025a).

Training details. We conduct RL training using the verl-tool (Jiang et al. 2025) frameworks on the Qwen-2.5-7B-Instruct model (Team 2024). The RL algorithm used is GRPO with the *clip-higher* trick, where the low and high clip ratios are set to 0.2 and 0.28, respectively. The learning rate is 1e-6, with a batch size of 1024, and training uses a maximum sequence length of 8192 tokens. The training setup includes a maximum of 5 interaction turns per episode, and feedback from <python> blocks is masked during loss computation. Key configuration parameters include a max prompt length of 9000 tokens, max response length of 3096 tokens, and max observation and action lengths of 2048 tokens, respectively.

Main Results

In-domain evaluation. As shown in Table 2, TAREX-7B achieves state-of-the-art performance across seven in-domain table reasoning tasks. It outperforms all competitors on these datasets, with its performance on TabFact (91.85) surpassing GPT-4o (92.09) and DeepSeek-R1 (90.25). Notably, TAREX-7B attains an average score of 79.92, surpassing GPT-4o (75.57) and DeepSeek-R1 (75.09) by substantial margins. It also outperforms recent RL-based table reasoning models such as Table-R1 (79.8) and Reasoning-Table-7B (75.46) by considerable margins across relevant tasks. Furthermore, unlike other methods that rely on supervised fine-tuning (SFT) or distillation techniques, TAREX does not use any form of SFT or distillation, yet still achieves superior performance.

Out-of-domain evaluation. TAREX-7B exhibits exceptional generalization in out-of-domain tests, achieving an average score of 69.33—1.85 points higher than GPT-4o (67.48) and 4.47 points above table-specific models (64.86) across RealHitBench, AIT-QA, TableBench, TabularGSM, OTT-QA, and Feverous. Unlike previous models, which are limited to short and simple tables in table reasoning, TAREX excels at handling complex tables.

Training curves. Figure 2 summarizes key training dynamics of TAREX on TaREx. The test score (Figure 2a) increases steadily on all three benchmarks, reaching strong performance by the end of training. Meanwhile, the answer format penalty quickly decays toward zero (Figure 2b), and the finish ratio stays consistently high and approaches 1.0 (Figure 2c), indicating stable rollouts and reliable output formatting. Notably, the mean number of actions per trajectory decreases over training (Figure 2d), suggesting improved decision efficiency. Finally, the count of filtered invalid trajectories (Figure 2e) grows in later training, reflecting increased exploration/complexity while the filtering mechanism actively removes invalid rollouts.

Ablation Study

In Table 2, we examine the roles of TAREX's three core contributions—unified action & observation space, code execution, and multi-turn interaction—through three ablation experiments with a progressive design. In Figure 3 and Table 4, we investigate the filtering mechanism used in data collection and RL training.

A unified DataFrame observation space improves structural understanding. As shown in Table 2, DF-Table-RL—using a DataFrame representation as the observation space—outperforms Table-RL with raw text input across all benchmarks. Performance gains are evident: +1.15 on WikiTQ (81.00 vs. 79.03), +1.70 on FinQA (69.61 vs. 67.91), and +1.17 on MultiHiertt (43.52 vs. 42.35). These results demonstrate the effectiveness of the DataFrame observation space in improving structural understanding. By enabling programmatic access, it reduces ambiguity in table interpretation, leading to better extraction and manipulation of data, particularly in complex datasets like WikiTQ, MultiHiertt, and FinQA.

Method	WikiTQ	MultiHiertt	FinQA	TAT-QA	HiTab	TabFact	MultimodalQA
Claude-3.7	82.02	40.98	57.45	53.09	75.96	91.12	68.23
GPT-4o	81.19	40.86	57.63	53.45	73.92	92.09	58.51
DeepSeek-R1	79.32	39.62	54.05	56.30	78.06	90.25	35.94
TableGPT2-7B	63.70	25.12	38.36	55.12	63.89	83.92	58.20
Qwen2.5-32B-Inst	79.65	37.74	59.2	67.29	73.29	89.48	56.33
Qwen2.5-7B-Inst	57.27	27.54	52.40	49.79	57.19	82.82	63.62
Table-R1	79.8	-	**70.8**	-	78.1	87.6	-
Reasoning-Table-7B	75.46	39.56	64.46	73.75	73.61	91.46	-
TAREX-7B	**83.62**	**47.32**	69.31	**80.37**	**80.33**	**91.85**	**78.18**
w/o MT (DF-Table-Code-RL)	81.00	43.52	69.61	78.97	79.23	90.76	77.43
w/o MT&CI (DF-Table-RL)	79.03	39.88	67.91	77.37	75.33	90.22	77.21
w/o MT& CI&DF (Table-RL)	78.26	39.09	66.41	77.68	73.43	89.96	76.08

Table 2: Performance on in-domain table reasoning tasks.

Method	AIT-QA	TableBench	TabularGSM	OTT-QA	Feverous	RealHitBench
Claude-3.7-Sonnet	89.73	62.52	86.20	62.69	71.84	75.88
GPT-4o	86.82	59.86	81.32	50.12	78.31	50.12
DeepSeek-R1	90.05	60.21	87.14	45.60	60.15	72.54
TableGPT2-7B	80.5	38.62	21.56	48.87	68.57	39.81
Qwen2.5-7B-Instruct	86.21	51.22	43.47	50.50	63.62	19.75
Reasoning-Table-7B	89.32	58.33	-	-	86.53	-
TAREX-7B	90.55	74.95	56.76	68.53	78.38	45.74

Table 3: Performance on out-of-domain table reasoning tasks.

Hybrid code-text actions enhance accuracy and reliability. DF-Table-Code-RL extends DF-Table-RL by incorporating a hybrid action space that combines Python code and natural language. This enhancement significantly improves performance across all tasks, as seen in Table 2. For instance, DF-Table-Code-RL improves FinQA by +1.42 (78.97 vs. 77.55), MultiHiertt by +1.08 (43.52 vs. 42.44), and TabFact by +2.19 (90.76 vs. 88.57), indicating that code execution enhances accuracy in table manipulation and arithmetic reasoning. The integration of code and natural language enables the model to handle more complex computations and longer tables, which is essential for tasks requiring extended contexts and intricate operations. This hybrid approach reduces hallucinations and enhances reliability, particularly for long-form reasoning and complex table-based queries.

Multi-Turn interaction enables stepwise reasoning and error correction. TAREX extends DF-Table-Code-RL with multi-turn interaction, achieving state-of-the-art performance on several tasks. For example, on WikiTQ, TAREX scores 84.12, outperforming DF-Table-Code-RL (81.00), and on FinQA, TAREX achieves 69.81 compared to 69.61. Other datasets, such as TAT-QA (80.87 vs. 78.97) and TabFact (92.35 vs. 90.76), also show significant improvements. Multi-turn interaction enables the model to decompose complex tasks into manageable steps, refine its reasoning iteratively, and correct intermediate errors. This stepwise approach is essential for solving intricate table reasoning tasks, where decisions depend on evolving contexts. The ability to correct errors during reasoning improves accuracy and enhances performance in tasks requiring sequential reasoning. This feature is particularly valuable for multi-turn tasks and complex table queries that demand iterative refinement and logical consistency.

Compact filtering mitigates optimization collapse and promotes efficient long-horizon reasoning. Figure 3 compares TAREX with its ablations in terms of training dynamics. Without compact filtering, training shows instability: response length increases significantly (often exceeding ˜3K), and the mean number of actions remains high, suggesting excessive resource use per rollout. Compact filtering, however, keeps response length controlled and reduces the action count, indicating more efficient decision-making. This results in more stable optimization signals—gradients stay non-degenerate and entropy loss evolves smoothly—indicating that compact filtering stabilizes training while promoting efficient long-horizon reasoning across steps.

Trajectory filtering stabilizes training by suppressing noisy and stagnant rollouts. Trajectory filtering removes unproductive rollouts with fragmented outputs or premature terminations, which introduce high-variance gradients. As shown in Figure 3, removing trajectory filtering increases gradient variance and leads to less controlled response lengths and actions. With trajectory filtering, these fluctuations de-

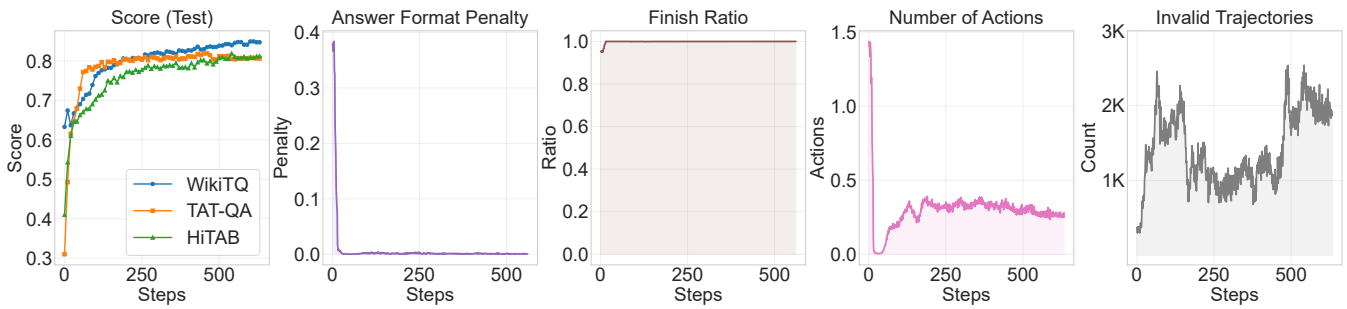

Figure 2: Trends of training-related metrics across three test datasets: (a) test score, (b) answer format penalty, (c) finish ratio, (d) number of actions, and (e) invalid trajectories.

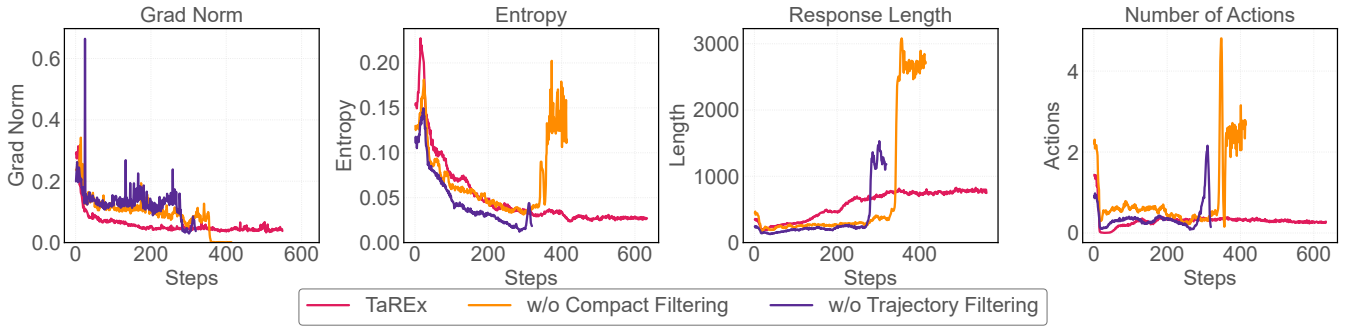

Figure 3: Trends in the number of turns, average response length, and performance scores (in-domain / out-of-domain) on the training dataset during RL training under trajectory and compact filtering mechanism.

crease, and the policy maintains more consistent rollout behavior. These results highlight the importance of trajectory filtering in stable multi-turn RL training, preventing the reinforcement of noisy behaviors and complementing compact filtering's role in managing the rollout budget.

Difficulty-based data filtering improves performance. We compare models trained with difficulty-filtered data to those trained on unfiltered data, keeping other RL settings unchanged. As shown in Table 4, models trained on filtered data outperform those on unfiltered data across all seven in-domain datasets, with gains ranging from +1.16 on WikiTQ to +1.49 on MultiHiertt. This indicates that difficulty-based filtering removes noisy samples while preserving valuable, semantically rich instances, enhancing model performance.

Case Study

Reasoning pattern analysis. As shown in Figure 4, In TAREX, three emergent multi-turn reasoning patterns manifest during interactive reasoning, illustrating the model's sophisticated problem-solving dynamics. *(1) Cross Validation.* Parallel execution of complementary reasoning branches to cross-validate intermediate outputs, bolstering result reliability via independent verification. *(2) Progressive Reasoning.* Hierarchical decomposition of complex tasks into sequential subtasks, where each step iteratively refines the reasoning chain using outputs from prior computations. *(3) Error Correction Loop.* Iterative adjustment mechanism that retries and modifies reasoning steps upon identifying errors, emulating human debugging processes to converge on accurate final results. Collectively, these patterns demonstrate TAREX's capacity to autonomously orchestrate multi-turn reasoning strategies, validating the effectiveness of its interactive framework in cultivating advanced problem-solving behaviors.

Performance across table length, quantity, and calculation scenarios. To verify TAREX's ability to handle long tables and calculation-intensive scenarios, we divided the test set into single/multi-table, long/short-table (using a 2K token threshold), and calculation/non-calculation categories. As shown in Figure 5, using Table-RL as a baseline, TAREX outperformed it in all settings. Notably, TAREX's advantage was most significant in the more challenging multi-table, long-table, and calculation scenarios, which validates the effectiveness of its multi-turn interactive reasoning. In simpler scenarios, its performance lead was less pronounced.

Related Work

Table reasoning. Table reasoning focuses on enabling models to understand and reason over structured tabular data to perform tasks such as question answering, fact verification, and text generation. Several datasets have been proposed, including those for table question answering (Zhong, Xiong, and Socher 2017; Pasupat and Liang 2015; Zhu et al. 2021; Chen et al. 2020, 2021b; Lei et al. 2023), fact verification (Chen et al. 2019; Aly et al. 2021), table-to-text (Parikh et al. 2020), text-to-SQL (Yu et al. 2018; Li et al. 2024) and data

	WikiTQ	MultiHiertt	FinQA	TAT-QA	HiTab	TabFact
Difficulty-filtered data	83.62	47.82	69.81	80.87	80.33	91.85
Unfiltered data	82.46$_{-1.16}$	46.33$_{-1.49}$	68.81$_{-1.00}$	79.79$_{-1.08}$	79.05$_{-1.28}$	90.49$_{-1.36}$

Table 4: Performance comparison on difficulty-filtered and unfiltered data.

Cross Validation (25.3%)

```
Code1:
df_rd = df[df['category'] == 'current $millions']
df_grouped = df_rd.groupby('Entity_Name')[['2010', '2011', '2012',
'2013', '2014', '2015']].mean()
result_1 = df_grouped
... ...
Code2:
df_pivot = df_rd.pivot(index='Entity_Name', columns='fiscal_year',
values='value')
df_pivot['avg_2010_2015'] =
df_pivot[[2010,2011,2012,2013,2014,2015]].mean(axis=1)
result_2 = df_pivot['avg_2010_2015']

Compare values in result_1 and result_2
```

Progressive Reasoning (47.8%)

```
# Step 1: Extract R&D data for 2010 and 2015
df_filtered = df[(df['category'] == 'current $millions') &
(df['fiscal_year'].isin([2010, 2015]))]
result_1 = df_filtered.pivot(index='Entity_Name',
columns='fiscal_year', values='value')
... ...

# Step 2: Calculating annual growth rate
result_1['cagr'] = (result_1[2015] / result_1[2010]) ** (1/5) - 1
result_2 = result_1[['cagr']]
... ...

# Step 3: Get top 10
top_10 = result_2.nlargest(10, 'cagr')
final_result = top_10.index.tolist()
```

Error Correction Loop (26.9%)

```
Initial Code (with Bug)
def calc_growth(df, entity):
   val_2015 = df[(df['Entity_Name'] == entity) &
(df['fiscal_year'] == 2015)]['value'].iloc[0]
   val_2010 = df[(df['Entity_Name'] == entity) &
(df['fiscal_year'] == 2011)]['value'].iloc[0]  # Error line
   return (val_2015 - val_2010) / val_2010

Error Detection: Domain knowledge (e.g., average industry growth
is 10-20%, but the result is 500%)

Corrected Code (Retry):
... ...
```

Figure 4: Case study on diverse reasoning patterns from TAREX, and we also present the frequency of occurrence of three distinct reasoning patterns in the test set.

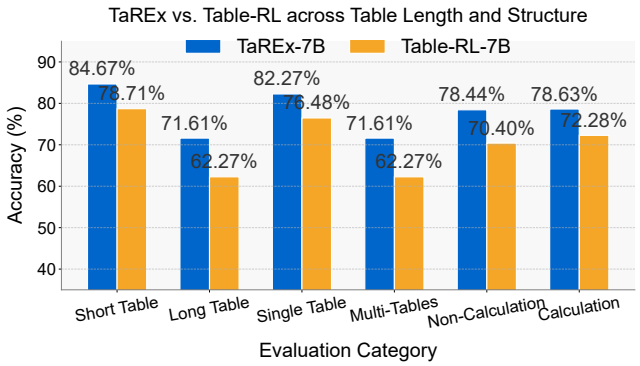

Figure 5: Experiments demonstrating the performance differences between TAREX and the Table-RL baseline across multi-/single-table, long-/short-table, and non-calculation/calculation examples.

science (Huang et al. 2024; Lei et al. 2025b), providing a wide range of datasets for the field. There are two main approaches: one involves carefully designed prompt engineering, leveraging large language models combined with tool calls like Python, SQL to address tasks (Cheng et al. 2022b; Jiang et al. 2023; Wang et al. 2024). The other approach uses fine-tuning with table-specific data to enable models to understand tables (Zhang et al. 2024a,b; Su et al. 2024). Reasoning-Table (Lei et al. 2025a), Table-R1 (Wu et al. 2025c; Yang et al. 2025) are the pioneering works to apply RLVR techniques to table reasoning. However, these existing RLVR methods for table reasoning do not incorporate code execution or multi-turn reasoning, which limits their reasoning capabilities. TAREX implements table reasoning based on multi-turn code execution and training, capable of handling more complex and realistic table reasoning scenarios.

RL for reasoning. The emergence of DeepSeek-R1 (Guo et al. 2025) shows RL's ability to guide models in generating high-quality reasoning trajectories. This has led to studies enhancing reasoning in mathematics (e.g., DeepScaleR (Luo et al. 2025b), DAPO (Yu et al. 2025)), and code generation (e.g., DeepCoder (Luo et al. 2025a)), with significant progress. The R1-paradigm has made breakthroughs in various NLP domains, such as tool use (Feng et al. 2025; Qian et al. 2025), RAG (Jin et al. 2025), and SQL generation (Pourreza et al. 2025; Ma et al. 2025). Some works have proposed using TIR combined with RL to solve tasks (Qian et al. 2025; Feng et al. 2025). TAREX focuses on code-integrated reinforcement learning for table reasoning, conducting comprehensive experiments and analysis.

Conclusion

We introduce TAREX, a reinforcement learning-based framework for table reasoning that tackles diverse table structures, complex reasoning, and multi-step tasks. TAREX integrates a unified executable table representation, a code-driven hybrid action space, and interactive multi-step reasoning, enhanced by rule-based rewards and trajectory filtering. TAREX-7B achieves state-of-the-art performance on in-domain and out-of-domain table reasoning benchmarks, outperforming proprietary LLMs and specialized models, especially on long and multi-table scenarios. These results highlight TAREX's robustness and scalability, paving the way for future advancements in table reasoning.

Acknowledgements

This work was supported by Beijing Natural Science Foundation (L243006) and the National Natural Science Foundation of China (No.62376270).

References

Aly, R.; Guo, Z.; Schlichtkrull, M.; Thorne, J.; Vlachos, A.; Christodoulopoulos, C.; Cocarascu, O.; and Mittal, A.

2021. Feverous: Fact extraction and verification over unstructured and structured information. *arXiv preprint arXiv:2106.05707*.

Anthropic, A. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 1: 1.

Borisov, V.; Leemann, T.; Seßler, K.; Haug, J.; Pawelczyk, M.; and Kasneci, G. 2022. Deep neural networks and tabular data: A survey. *IEEE transactions on neural networks and learning systems*.

Cao, B.; Lin, H.; Han, X.; and Sun, L. 2024. The Life Cycle of Knowledge in Big Language Models: A Survey. *Machine Intelligence Research*, 21(2): 217–238.

Chen, W.; Chang, M.-W.; Schlinger, E.; Wang, W. Y.; and Cohen, W. W. 2021a. Open Question Answering over Tables and Text. In *International Conference on Learning Representations*.

Chen, W.; Wang, H.; Chen, J.; Zhang, Y.; Wang, H.; Li, S.; Zhou, X.; and Wang, W. Y. 2019. TabFact: A Large-scale Dataset for Table-based Fact Verification. In *International Conference on Learning Representations*.

Chen, W.; Zha, H.; Chen, Z.; Xiong, W.; Wang, H.; and Wang, W. Y. 2020. HybridQA: A Dataset of Multi-Hop Question Answering over Tabular and Textual Data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 1026–1036.

Chen, Z.; Chen, W.; Smiley, C.; Shah, S.; Borova, I.; Langdon, D.; Moussa, R.; Beane, M.; Huang, T.-H.; Routledge, B. R.; et al. 2021b. FinQA: A Dataset of Numerical Reasoning over Financial Data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 3697–3711.

Cheng, Z.; Dong, H.; Wang, Z.; Jia, R.; Guo, J.; Gao, Y.; Han, S.; Lou, J.-G.; and Zhang, D. 2022a. 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)*, 1094–1110.

Cheng, Z.; Hao, S.; Liu, T.; Zhou, F.; Xie, Y.; Yao, F.; Bian, Y.; Zhuang, Y.; Dey, N.; Zha, Y.; et al. 2025. Revisiting Reinforcement Learning for LLM Reasoning from A Cross-Domain Perspective. *arXiv preprint arXiv:2506.14965*.

Cheng, Z.; Xie, T.; Shi, P.; Li, C.; Nadkarni, R.; Hu, Y.; Xiong, C.; Radev, D.; Ostendorf, M.; Zettlemoyer, L.; et al. 2022b. Binding language models in symbolic languages. *arXiv preprint arXiv:2210.02875*.

Chu, T.; Zhai, Y.; Yang, J.; Tong, S.; Xie, S.; Schuurmans, D.; Le, Q. V.; Levine, S.; and Ma, Y. 2025. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*.

Feng, J.; Huang, S.; Qu, X.; Zhang, G.; Qin, Y.; Zhong, B.; Jiang, C.; Chi, J.; and Zhong, W. 2025. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*.

Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Huang, Y.; Luo, J.; Yu, Y.; Zhang, Y.; Lei, F.; Wei, Y.; He, S.; Huang, L.; Liu, X.; Zhao, J.; et al. 2024. DA-Code: Agent Data Science Code Generation Benchmark for Large Language Models. *arXiv preprint arXiv:2410.07331*.

Jiang, D.; Lu, Y.; Li, Z.; Lyu, Z.; Nie, P.; Wang, H.; Su, A.; Chen, H.; Zou, K.; Du, C.; et al. 2025. Verltool: Towards holistic agentic reinforcement learning with tool use. *arXiv preprint arXiv:2509.01055*.

Jiang, J.; Zhou, K.; Dong, Z.; Ye, K.; Zhao, X.; and Wen, J.-R. 2023. StructGPT: A General Framework for Large Language Model to Reason over Structured Data. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 9237–9251. Singapore: Association for Computational Linguistics.

Jin, B.; Zeng, H.; Yue, Z.; Yoon, J.; Arik, S.; Wang, D.; Zamani, H.; and Han, J. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Katsis, Y.; Chemmengath, S.; Kumar, V.; Bharadwaj, S.; Canim, M.; Glass, M.; Gliozzo, A.; Pan, F.; Sen, J.; Sankaranarayanan, K.; et al. 2022. AIT-QA: Question Answering Dataset over Complex Tables in the Airline Industry. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, 305–314.

Lei, F.; Chen, J.; Ye, Y.; Cao, R.; Shin, D.; Hongjin, S.; SUO, Z.; Gao, H.; Hu, W.; Yin, P.; et al. 2024. Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows. In *The Thirteenth International Conference on Learning Representations*.

Lei, F.; Liu, Q.; Huang, Y.; He, S.; Zhao, J.; and Liu, K. 2023. S3Eval: A Synthetic, Scalable, Systematic Evaluation Suite for Large Language Models. *arXiv preprint arXiv:2310.15147*.

Lei, F.; Meng, J.; Huang, Y.; Chen, T.; Zhang, Y.; He, S.; Zhao, J.; and Liu, K. 2025a. Reasoning-table: Exploring reinforcement learning for table reasoning. *arXiv preprint arXiv:2506.01710*.

Lei, F.; Meng, J.; Huang, Y.; Zhao, J.; Zhang, Y.; Luo, J.; Zou, X.; Yang, R.; Shi, W.; Gao, Y.; et al. 2025b. DAComp: Benchmarking Data Agents across the Full Data Intelligence Lifecycle. *arXiv preprint arXiv:2512.04324*.

Li, J.; Hui, B.; Qu, G.; Yang, J.; Li, B.; Li, B.; Wang, B.; Qin, B.; Geng, R.; Huo, N.; et al. 2024. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36.

Luo, M.; Tan, S.; Huang, R.; Patel, A.; Ariyak, A.; Wu, Q.; Shi, X.; Xin, R.; Cai, C.; Weber, M.; Zhang, C.; Li, L. E.; Popa, R. A.; and Stoica, I. 2025a. DeepCoder: A Fully Open-Source 14B Coder at O3-mini Level. Notion Blog.

Luo, M.; Tan, S.; Wong, J.; Shi, X.; Tang, W. Y.; Roongta, M.; Cai, C.; Luo, J.; Li, L. E.; Popa, R. A.; and Stoica, I. 2025b. DeepScaleR: Surpassing O1-Preview with a 1.5B Model by Scaling RL. Notion Blog.

Ma, P.; Zhuang, X.; Xu, C.; Jiang, X.; Chen, R.; and Guo, J. 2025. SQL-R1: Training Natural Language to SQL Reasoning Model By Reinforcement Learning. *arXiv preprint arXiv:2504.08600*.

OpenAI, R. 2023. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2: 13.

Parikh, A.; Wang, X.; Gehrmann, S.; Faruqui, M.; Dhingra, B.; Yang, D.; and Das, D. 2020. ToTTo: A Controlled Table-To-Text Generation Dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1173–1186.

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

Pourreza, M.; Talaei, S.; Sun, R.; Wan, X.; Li, H.; Mirhoseini, A.; Saberi, A.; Arik, S.; et al. 2025. Reasoning-SQL: Reinforcement Learning with SQL Tailored Partial Rewards for Reasoning-Enhanced Text-to-SQL. *arXiv preprint arXiv:2503.23157*.

Qian, C.; Acikgoz, E. C.; He, Q.; Wang, H.; Chen, X.; Hakkani-Tür, D.; Tur, G.; and Ji, H. 2025. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*.

Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Bi, X.; Zhang, H.; Zhang, M.; Li, Y.; Wu, Y.; et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Su, A.; Wang, A.; Ye, C.; Zhou, C.; Zhang, G.; Chen, G.; Zhu, G.; Wang, H.; Xu, H.; Chen, H.; et al. 2024. TableGPT2: A Large Multimodal Model with Tabular Data Integration. *CoRR*.

Talmor, A.; Yoran, O.; Catav, A.; Lahav, D.; Wang, Y.; Asai, A.; Ilharco, G.; Hajishirzi, H.; and Berant, J. 2021. MultiModalQA: complex question answering over text, tables and images. In *International Conference on Learning Representations*.

Team, Q. 2024. Qwen2.5: A Party of Foundation Models.

Tian, S.-Y.; Zhou, Z.; Dong, W.; Yang, M.; Yu, K.-Y.; Cheng, Z.-J.; Guo, L.-Z.; and Li, Y.-F. 2025. Automated Text-to-Table for Reasoning-Intensive Table QA: Pipeline Design and Benchmarking Insights. *arXiv preprint arXiv:2505.19563*.

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. 2024. Chain-of-Table: Evolving Tables in the Reasoning Chain for Table Understanding. In *The Twelfth International Conference on Learning Representations*.

Wu, P.; Yang, Y.; Zhu, G.; Ye, C.; Gu, H.; Lu, X.; Xiao, R.; Bao, B.; He, Y.; Zha, L.; et al. 2025a. RealHiTBench: A Comprehensive Realistic Hierarchical Table Benchmark for Evaluating LLM-Based Table Analysis. *arXiv preprint arXiv:2506.13405*.

Wu, X.; Yang, J.; Chai, L.; Zhang, G.; Liu, J.; Du, X.; Liang, D.; Shu, D.; Cheng, X.; Sun, T.; et al. 2025b. Tablebench: A comprehensive and complex benchmark for table question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 25497–25506.

Wu, Z.; Yang, J.; Liu, J.; Wu, X.; Pan, C.; Zhang, J.; Zhao, Y.; Song, S.; Li, Y.; and Li, Z. 2025c. Table-R1: Region-based Reinforcement Learning for Table Understanding. *arXiv preprint arXiv:2505.12415*.

Xue, Z.; Zheng, L.; Liu, Q.; Li, Y.; Zheng, X.; Ma, Z.; and An, B. 2025. Simpletir: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. *arXiv preprint arXiv:2509.02479*.

Yang, Z.; Chen, L.; Cohan, A.; and Zhao, Y. 2025. Table-R1: Inference-Time Scaling for Table Reasoning. *arXiv preprint arXiv:2505.23621*.

Yu, Q.; Zhang, Z.; Zhu, R.; Yuan, Y.; Zuo, X.; Yue, Y.; Fan, T.; Liu, G.; Liu, L.; Liu, X.; et al. 2025. DAPO: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Yu, T.; Zhang, R.; Yang, K.; Yasunaga, M.; Wang, D.; Li, Z.; Ma, J.; Li, I.; Yao, Q.; Roman, S.; et al. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 3911–3921.

Zha, L.; Zhou, J.; Li, L.; Wang, R.; Huang, Q.; Yang, S.; Yuan, J.; Su, C.; Li, X.; Su, A.; et al. 2023. Tablegpt: Towards unifying tables, nature language and commands into one gpt. *arXiv preprint arXiv:2307.08674*.

Zhang, T.; Yue, X.; Li, Y.; and Sun, H. 2024a. TableLlama: Towards Open Large Generalist Models for Tables. In Duh, K.; Gomez, H.; and Bethard, S., eds., *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 6024–6044. Mexico City, Mexico: Association for Computational Linguistics.

Zhang, X.; Luo, S.; Zhang, B.; Ma, Z.; Zhang, J.; Li, Y.; Li, G.; Yao, Z.; Xu, K.; Zhou, J.; et al. 2024b. Tablellm: Enabling tabular data manipulation by llms in real office usage scenarios. *arXiv preprint arXiv:2403.19318*.

Zhang, Y.; song Tang, X.; and Hao, K. 2025. CRMR: A Collaborative Multistep Reasoning Framework for Solving Mathematical Problems. *Machine Intelligence Research*, 22(3): 571–584.

Zhao, Y.; Li, Y.; Li, C.; and Zhang, R. 2022. MultiHiertt: Numerical Reasoning over Multi Hierarchical Tabular and Textual Data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 6588–6600.

Zhong, V.; Xiong, C.; and Socher, R. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Zhu, F.; Lei, W.; Huang, Y.; Wang, C.; Zhang, S.; Lv, J.; Feng, F.; and Chua, T.-S. 2021. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. *arXiv preprint arXiv:2105.07624*.