Table-R1:Region-based Reinforcement Learning for Table Understanding

Zhenhe Wu[1,2†], Jian Yang[1*], Zhongjiang He[2], Changzai Pan[2], Jiaheng Liu[1], Xianjie Wu[1], Yu Zhao[2], Shuangyong Song[2], Yongxiang Li[2], Zhoujun Li[1], Xuelong Li[3*],
[1]Beihang University,
[2]Xingchen AGI Lab, China Telecom Artificial Intelligence Technology (Beijing) Co., Ltd,
[3]Institute of Artificial Intelligence (TeleAI), China Telecom,
{wuzhenhe,jiaya}@buaa.edu.cn
xuelong_li@ieee.org

Abstract

Tables present unique challenges for language models due to their structured row-column interactions, necessitating specialized approaches for effective comprehension. While large language models (LLMs) have demonstrated potential in table reasoning through prompting and techniques like chain-of-thought (CoT) and program-of-thought (PoT), optimizing their performance for table question answering remains underexplored. In this paper, we introduce region-based Table-R1, a novel reinforcement learning approach that enhances LLM table understanding by integrating region evidence into reasoning steps. We employ Region-Enhanced Supervised Fine-Tuning (RE-SFT) to guide models in identifying relevant table regions before generating answers, incorporating textual, symbolic, and program-based reasoning. Additionally, Table-Aware Group Relative Policy Optimization (TARPO) introduces a mixed reward system to dynamically balance region accuracy and answer correctness, with decaying region rewards and consistency penalties to align reasoning steps. Experiments show that Table-R1 achieves an average performance improvement of 14.36 points across multiple base models on three benchmark datasets, while TARPO significantly reduces the reasoning token consumption by 67.5% compared to GRPO, significantly advancing LLM capabilities in efficient tabular reasoning.

1 Introduction

Tables are a widely used data format different from plain text, which poses unique challenges for language models due to their structured row-column interactions (Wang et al., 2024d; Sui et al., 2024). Understanding tabular data is crucial for applications like fact verification and question answering, driving significant research interest. Unlike plain text, tables rely on complex row-column interactions, making them challenging for models to interpret. Previous researchers have developed specialized embedding layers, attention mechanisms, and pre-training objectives to enhance structural awareness.

The rise of large language models (LLMs), such as general LLMs (OpenAI, 2023, 2025; Hurst et al., 2024; Wang et al., 2024b,a, 2025; Liu et al., 2025a), has introduced new opportunities for table understanding, as their massive pre-training enables strong performance through prompting alone. Techniques like chain-of-thought (Wei et al., 2022) (CoT) and program-of-thought (Chen et al., 2022) (PoT) have further enhanced the reliability of LLMs by equipping responses with reasoning steps. The reasoning LLMs, such as o1/o3 (Guo et al., 2025a; Jaech et al., 2024), introduce the step-by-step reasoning trajectories to boost test-time accuracy, which is optimized by reinforcement learning (RL). Deepseek-R1 (Guo et al., 2025a) combines group relative policy optimization (Shao et al., 2024) (GRPO) and a rule-based reward system to effectively improve the performance of coding and other tasks. Besides, a comprehensive and complex table question answering benchmark TableBench (Wu et al., 2024) is proposed to evaluate table reasoning capabilities of LLMs. However, prior studies often relied on general LLM capabilities and standard training paradigms without proposing training methods specifically tailored to the unique structural constraints of tables.

To unleash the potential of LLMs in table understanding, we explore the approach of rule-based reinforcement learning to significantly strengthen table understanding capabilities by introducing the table structure information (Nahid and Rafiei, 2024; Ye et al., 2023; Cheng et al., 2023). We propose a region-based table question answering LLM (Table-R1) by injecting the region evidence into the reasoning process and then facilitating the LLM to infer

*Corresponding author.
†Work done during the internship at China Telecom Artificial Intelligence Technology (Beijing) Co., Ltd.

Findings of the Association for Computational Linguistics: ACL 2026, pages 27374–27391

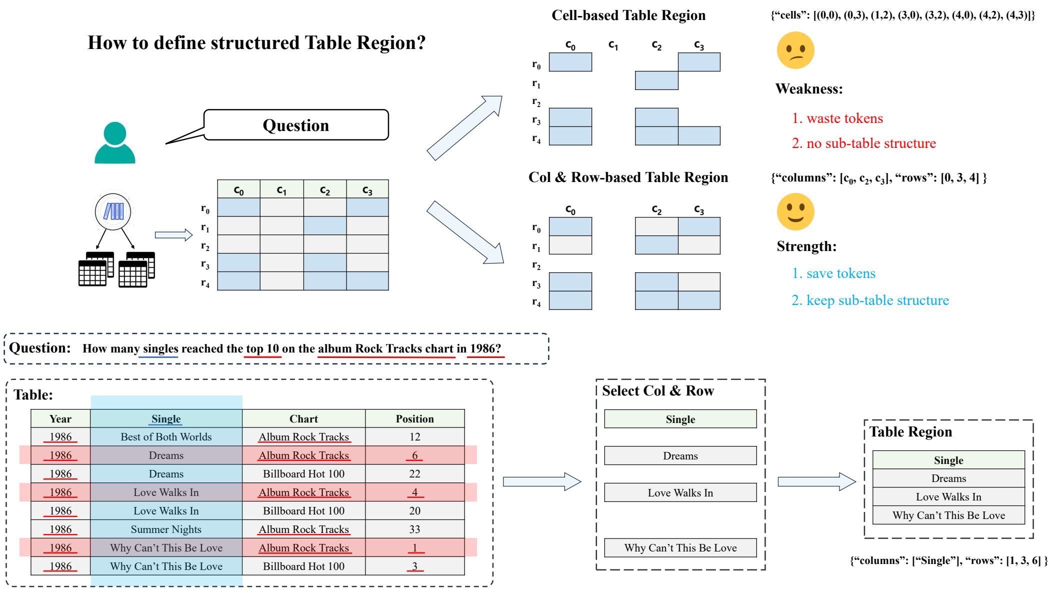


Figure 1: In Table-R1, we adopt the col & row-based table region for its structured definition. Compared to the cell-based Table Region, it not only saves input tokens but also preserves the sub-table structure. The specific example demonstrates how the LLM can extract Table Region.

the correct answer through reinforcement learning. Specifically, Table-R1 is first trained with the RE-SFT (Region-Enhanced Supervised Fine-Tuning) and then enhanced by TARPO (Table-Aware Group Relative Policy Optimization). RE-SFT enhances supervised fine-tuning by guiding LLMs to first identify relevant table regions before generating answers, integrating this step into three reasoning processes, including textual chain-of-thought (TCoT), symbolic chain-of-thought (SCoT), and program-of-thoughts (PoT). TARPO extends reinforcement learning by introducing a mixed reward system that dynamically balances table region accuracy and answer correctness, with a consistency preference to ensure alignment between region identification and answer generation. By decaying the region reward weight over time and penalizing optimization inconsistencies, Table-R1 effectively optimizes both region identification and answer quality, significantly improving table understanding performance.

Table-R1 gets the top-tier performance on different benchmarks by a large margin, which demonstrates the effectiveness of our region-based table reasoning method. Our contributions and findings can be summarized as follows:

1. We propose a specialized LLM reinforcement training approach designed to exploits tabular structures, integrating table regions into reasoning trajectories and designing reward functions based on both table region predictions and final answers correctness.

2. We present Table-R1, a unified framework that integrates RE-SFT and TARPO to synergistically guide LLMs in table region selection and answer generation across both supervised fine-tuning and reinforcement learning phases.

3. Our experimental results show that the Table-R1 framework delivers significant improvements in performance and generalization across various benchmarks, achieving an average performance gain of 14.36 points while reducing the reasoning tokens by 67.5%. Furthermore, we introduce the TableInstruct-RE dataset, which enhances Chain-of-Thought (CoT) reasoning with explicit table region annotations derived from TableInstruct.

2 Method

2.1 Problem Definition

Table question answering (Table QA) is defined as: Given a semi-structured table $\mathcal{T}$ with R rows and C columns, the objective is to generate an answer $\mathcal{A}$ to a question $\mathcal{Q}$ by leveraging the information within $\mathcal{T}$. Here, $\mathcal{A}$ is a set of values or entities expressed as $\{a_1, a_2, \ldots, a_k\}$ $(k \in \mathbb{N}^+)$.

2.2 Table Region Definition

To concisely and intuitively describe a structured Table Region, we define it as: T_{reg} $=$

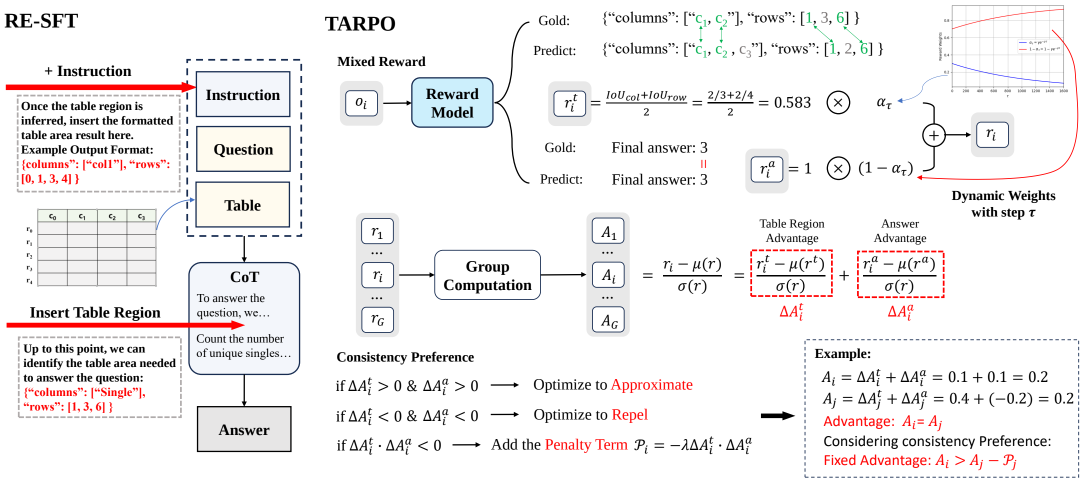


Figure 2: The framework of Table-R1. In RE-SFT, we integrate the minimum table region at the correct position in the CoT process and add relevant instructions in the LLM prompt. In TARPO, we introduce mixed reward to balance the table region result and the answer result, and use α_τ to dynamically adjust the weights in training.

$\{[c_1, c_2, ..., c_i], [r_1, r_2, .., r_j]\}$. Figure 1 illustrates why we use col & row-based structure and present an example of how the LLM extracts relevant table region. Compared to the cell-based structure, col & row-based structure not only saves input tokens but also preserves the sub-table structure. Though it may inevitably include some irrelevant cells, it still significantly narrows the data-acquisition scope for the LLM when computing answers. In the specific example, by combining the column attributes and cell values of the table with the target question, LLM can select the minimal table region required to generate the final answer. This approach efficiently narrows down the relevant data, focusing on the "Single" column and the corresponding rows that match the query criteria.

2.3 Region-Enhanced Supervised Fine-Tuning

In this section, we present **RE-SFT**, a Region-Enhanced Supervised Fine-Tuning approach. It aims to steer LLMs toward first identifying the table region relevant to answering questions during the CoT process. Subsequently, by leveraging the data from these identified table regions, the LLMs can compute and arrive at the final answers. We formalize the ICL (In-context Learning) process as $\mathcal{M}(\mathcal{I}(\mathcal{T}, \mathcal{Q}), \mathcal{E})$, where $\mathcal{I}$ denotes the task-specific instruction, $\mathcal{E}$ represents a few output examples, and $\mathcal{M}$ refers to the LLM.

Tablebench (Wu et al., 2024) has summarized four reasoning methods in TableQA: Direct Prompting (DP), Textual Chain-of-Thought (TCoT), Symbolic Chain-of-Thought (SCoT), and Program-of-Thoughts (PoT). We integrate the minimal table region T_{reg} into these four methods:

$$DP : \mathcal{M}(\mathcal{I}_D(\mathcal{T}, \mathcal{Q}), \mathcal{E}) \rightarrow \{T_{reg}, \mathcal{A}\} \tag{1}$$

$$TCoT : \mathcal{M}(\mathcal{I}_T(\mathcal{T}, \mathcal{Q}), \mathcal{E}) \rightarrow \{r_1, r_2, \ldots, r_m, T_{reg}, r_{m+1}, \ldots, r_k, \mathcal{A}\} \tag{2}$$

$$SCoT : \mathcal{M}(\mathcal{I}_S(\mathcal{T}, \mathcal{Q}), \mathcal{E}) \rightarrow \{(r_{a_1}, r_{p_1}, r_{s_1}), \ldots, T_{reg}, \ldots, (r_{a_k}, r_{p_k}, r_{s_k}), \mathcal{A}\} \tag{3}$$

$$PoT : \mathcal{M}(\mathcal{I}_P(\mathcal{T}, \mathcal{Q}), \mathcal{E}) \rightarrow \{T_{reg}, \mathcal{P}\} \rightarrow \mathcal{A} \tag{4}$$

where r_k is the k-th reasoning step in TCoT. r_{a_k} is the analyzing step, r_{p_k} is the program commands generating step, and r_{s_k} is the result simulation step in SCoT. $\mathcal{P}$ is the generated code in PoT.

The training objective $\mathcal{L}_{\text{all}}$ of RE-SFT on datasets containing all four reasoning methods:

$$\mathcal{L}_{\text{all}} = -\sum_{n=1}^{N} \mathbb{E}_{q^{R_n}, a^{R_n}} \left[\log P(a^{R_n} | q^{R_n}; \mathcal{M})\right] \tag{5}$$

where q^{R_n} and a^{R_n} denote the table-related question and answer from dataset D^{R_n} of reasoning method R_n, respectively. N represents the total number of reasoning methods.

2.4 Table-Aware Reinforcement Learning

Recently, reinforcement learning has shown remarkable performance on numerous tasks. In the specific tableQA task, we intend to offer additional rewards to the reasoning process that accurately identifies the relevant table region for better answer generation. Therefore, we have extended GRPO to develop the **TARPO** (Table-Aware Group Relative Policy Optimization) algorithm, which provides joint incentives for both the correctness of table region identification and final answer generation.

2.4.1 Mixed Reward

The reward r_i for reasoning process i is obtained by weighting the sum of the table region reward r_i^t and the answer reward r_i^a:

$$r_i = \alpha_\tau r_i^t + (1 - \alpha_\tau) r_i^a \tag{6}$$

where r_i^a is binary, taking a value of 1 when the inferred answer matches the correct answer, and 0 otherwise (Numerical answers match if equal, while string answers match if the Rouge-L score exceeds threshold ζ). r_i^t is calculated by computing the IoU (Intersection over Union) (Redmon and Farhadi, 2017) separately for the rows and columns between the table region obtained in the reasoning process and the ground truth. The results are then summed and averaged, with $r_i^t \in [0, 1]$:

$$r_i^t = \frac{IoU_{col}(T_{reg}, \hat{T_{reg}}) + IoU_{row}(T_{reg}, \hat{T_{reg}})}{2} \tag{7}$$

where IOU_{col} and IOU_{row} respectively denote the IOU functions specific to rows and columns. $\hat{T_{reg}}$ stands for the predicted table region, while T_{reg} represents the ground truth of the table region.

2.4.2 Dynamic Weight of Reward

At the beginning of reinforcement learning, our goal is to capture table regions more accurately to assist with answer reasoning. However, as training progresses and the model acquires a certain level of table region selection capability, we shift the focus of the reward more towards the accuracy of the answers. Thus, we have designed a dynamically changing weight α_τ , which decreases with increasing training step τ, the formula is as follows:

$$\alpha_\tau = \gamma e^{-\rho\tau} \tag{8}$$

where $\gamma \in [0, 1]$ is the initial weight, $\rho > 0$ is the decay coefficient.

2.4.3 Consistency Preference

Following GRPO, we can obtain the normalized group reward score A_i by calculating the variance and standard deviation of the rewards. Combining equation (6), we can further decompose A_i into two components ΔA_i^t and ΔA_i^a:

$$\begin{aligned} A_i =& \frac{r_i - \text{mean}(\{r_1, r_2, \ldots, r_G\})}{\text{std}(\{r_1, r_2, \ldots, r_G\})} \\ =& \underbrace{\frac{\alpha_\tau [r_i^t - \text{mean}(\{r_1^t, r_2^t, \ldots, r_G^t\})]}{\text{std}(\{r_1, r_2, \ldots, r_G\})}}_{\Delta A_i^t} + \\ & \underbrace{\frac{(1-\alpha_\tau)[r_i^a - \text{mean}(\{r_1^a, r_2^a, \ldots, r_G^a\})]}{\text{std}(\{r_1, r_2, \ldots, r_G\})}}_{\Delta A_i^a} \end{aligned} \tag{9}$$

Since the correct table region can guide better answer generation, we prefer the model to optimize in the direction where both ΔA_i^t and ΔA_i^a are positive. If ΔA_i^t and ΔA_i^a are of opposite signs, we consider that the reasoning process might be accidental and violates the consistency preference. Therefore, we introduce a small penalty $\mathcal{P}_i$ to mitigate the tendency of optimizing towards strategies (or accelerate the optimization away from it) that do not satisfy consistency:

$$\mathcal{P}_i = \begin{cases} 0 & \text{if } \Delta A_i^t \cdot \Delta A_i^a > 0 \\ -\lambda \Delta A_i^t \cdot \Delta A_i^a & \text{if } \Delta A_i^t \cdot \Delta A_i^a < 0 \end{cases} \tag{10}$$

Incorporating the penalty term $\mathcal{P}_i$, the objective of TARPO is:

$$\begin{aligned} J_{\text{TARPO}}(\theta) = \mathbb{E}_{q, \{o_i\}_{i=1}^G} \Bigg[\frac{1}{G} \sum_{i=1}^{G} \min \Bigg(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)} \\ (A_i - \mathcal{P}_i), \text{clip} \left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon \right) \\ (A_i - \mathcal{P}_i) \Bigg) - \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) \Bigg] \end{aligned} \tag{11}$$

3 Experimental Setup

Datasets. We use **TableInstruct** (Wu et al., 2024) as the training set because it encompasses all four reasoning methods, includes four major and 18 sub-question categories with the total size of nearly 20,000. For testing purposes, we employed three benchmark datasets. **TableBench** (Wu et al., 2024) is a comprehensive tabular benchmark designed to evaluate large language models across multiple table-related tasks. **WikiTQ** (Pasupat and Liang, 2015a) is a dataset for table question answering, featuring 22,033 question-answer pairs with 2,108 tables. **WikiSQL** (Zhong et al., 2017) is a dataset that annotates Wiki tables with SQL, which contains 81,000 questions and 24,000 tables. The dataset statistics are summarized in Table 1.

Dataset	Tables	Samples	Input Token Length		
			min	max	median
TableInstruct	1.1K	19,661	242	7,539	755
TableBench	0.59K	3,544	324	8,192	776
WikiTQ (test)	0.4K	4,344	266	2,120	691
WikiSQL (test)	5K	15,878	204	2,034	566

Table 1: Statistics of datasets.

Evaluation Metrics. We utilize the official evaluation metrics and codes of each benchmark dataset. For TableBench, we apply a combination of **Exact Match (EM)**, **Rouge-L** (Lin, 2004) and **Pass@1** (Chen et al., 2021) metrics, while for WikiTQ and WikiSQL, we evaluate **Accuracy**.

Baselines. We select several competitive LLMs similar in size as baselines. The first category is composed of the most advanced open-source general-purpose LLMs, including DeepSeek-Coder-V2-Lite-16B (DeepSeek-AI et al., 2024), Yi Coder-9B-Chat (01.AI, 2024), Qwen2.5-Coder-7B-Instruct (Hui et al., 2024), Qwen2.5-7B-Instruct (Yang et al., 2025b) and QWQ-32B (Yang et al., 2025a). We also use GPT-4o (Hurst et al., 2024) as a powerful baseline. The second category comprises LLMs specifically designed for table-related tasks. TableLLM-13B (Zhang et al., 2024c) fine-tuned CodeLlama-13B (Roziere et al., 2023) to handle a variety of table operations in spreadsheet and document settings. CHAIN-OF-TABLE (Wang et al., 2024c) represents a tabular reasoning chain through in-context learning. TableGPT2-7B (Su et al., 2024) specifically designed a novel table encoder to capture schema-level and cell-level information. To better illustrate the improvement of Table-R1, we introduce a third category of same-base LLMs that use the identical TableInstruct as the training set, including TableLLMs (Wu et al., 2024) fine-tuned on Qwen2-7B (Yang et al., 2024a), CodeQwen-7B (Bai et al., 2023), Deepseek-Coder-7B (Guo et al., 2024), Llama3.1-8B (Dubey et al., 2024), TeleChat2 (Wang et al., 2025), Qwen2.5-3B (Yang et al., 2025b), and Qwen3-8B (Yang et al., 2025a).

Implementation Details. (1) For Datasets: We employ DeepSeek-R1 to generate the minimal table regions that suffice for each critical reasoning step in the original TableInstruct dataset, manually verify the correctness, and obtain the **TableInstruct-RE** dataset. We also add instructions related to table region generation in the prompts of both the training and test sets. As WikiSQL and WikiTQ lack ICL prompts, we use the modified TCoT prompt template of TableBench for them. **(2) For RE-SFT:** We achieve supervised fine-tuning on TableInstruct-RE dataset. Following the same setting of TableBench(Wu et al., 2024), we use the entire dataset for training purposes and do not partition a validation set. We utilize a cosine annealing scheduler, which sets the initial learning rate at $2e^{-5}$, and conduct training over 3 epochs. Optimization is performed using the Adam optimizer, with a batch size of 512 and a maximum sequence length of 4096. We use 8*A100 GPUs for training, which takes 2 to 2.5 hours for each training. **(3) For TARPO:** We randomly shuffled the TableInstruct-RE dataset and split it into training and validation sets in a ratio of 9:1. We set the batch size to 32, with the max prompt length and max response length at 8192 and 2048. The learning rate is $7e^{-7}$, and the group number G for each question is 16. The 7B/8B-sized LLMs are trained on 4*80GB-H100 GPUs, and the 3B-sized LLM on 8*40GB-A100 GPUs. The training lasts for 3 epochs and around 72 hours. For the hyper parameters, we set $\{\zeta, \gamma, \rho, \lambda\}$ to $\{0.6, 0.3, 9e^{-4}, 0.1\}$.

4 Results and Analysis

4.1 Overall Results

Table 2 shows the TableBench, WikiTQ, and WikiSQL results for baselines and our Table-R1. We calculate an overall score from the average test-set scores to evaluate LLM performance. Among general-purpose LLMs, the small-sized (under 16B) Qwen2.5 series shows strong overall performance, with Qwen2.5-Coder scoring 38.07. Specifically, the Qwen2.5-Coder achieves the overall score of 38.07. Larger models like GPT-4o achieve a higher overall score of 49.17 and perform well across various datasets. For table-focused LLMs, TableGPT2-7B model stands out in small models with a 42.2 overall score, surpassing general-purpose LLMs and TableLLMs of the same size. The larger TableGPT2-72B further boosts the overall score to 46.2.

We implement the Table-R1 method across different LLM architectures and compare its performance with TableLLMs trained on the same TableInstruct training dataset. Each experimental group consisted of three model configurations: 1) the baseline TableLLM, 2) RE-SFT introduced via supervised fine-tuning, and 3) the model with both RE-SFT and TARPO (Table-R1). This design sys-

Model	Base Model	Size	TableBench				WikiTQ	WikiSQL	Overall
			DP	TCoT	SCoT	PoT			
General-purpose LLMs									
Yi-Coder	Yi	9B	21.94	22.80	8.43	11.36	43.37	25.34	22.21
DS-lite	DS-Coder	16B	29.60	30.93	22.61	10.90	47.65	38.30	30.00
Qwen2.5-Instruct	Qwen2.5	7B	25.18	29.77	24.35	22.58	68.55	47.42	36.14
Qwen2.5-Coder	Qwen2.5	7B	28.67	36.25	25.95	16.15	74.50	46.90	38.07
QWQ	Qwen2.5	32B	43.87	43.48	37.06	31.58	70.50	47.67	45.69
GPT-4o	-	-	40.91	51.96	41.43	45.71	68.40	47.60	49.17
Table-focused LLMs									
TableLLM	CodeLlama	13B	3.88	3.85	2.88	2.94	66.30	41.10	20.16
CHAIN-OF-TABLE	GPT3.5	-	24.86	32.61	26.39	20.24	59.94	43.72	34.63
TableGPT2	Qwen2.5	7B	27.95	41.05	31.4	38.67	61.42	53.74	42.20
TableGPT2	Qwen2.5	72B	38.90	50.06	30.47	28.98	71.45	57.32	46.20
The same-base LLMs training on TableInstruct									
TableLLM (w/ SFT)			22.29	31.90	23.62	12.87	**64.16**	46.50	33.56
RE-TableLLM (w/ RE-SFT)	Qwen2	7B	28.24	37.33	29.10	41.05	57.55	64.42	42.95
Table-R1 (w/ RE-SFT & TARPO)			**36.22**	**41.85**	**32.84**	**41.13**	61.69	**64.80**	**46.42**
TableLLM (w/ SFT)			20.15	24.81	20.55	15.14	36.05	37.20	25.65
RE-TableLLM (w/ RE-SFT)	CodeQwen	7B	21.47	26.49	24.34	**42.52**	46.71	57.35	36.48
Table-R1 (w/ RE-SFT & TARPO)			**26.63**	**28.42**	**26.10**	41.15	**49.31**	**59.89**	**38.58**
TableLLM (w/ SFT)			23.15	30.51	23.56	18.74	36.05	36.14	25.99
RE-TableLLM (w/ RE-SFT)	DS-Coder	7B	25.63	31.49	25.88	41.63	47.88	58.86	36.13
Table-R1 (w/ RE-SFT & TARPO)			**28.74**	**35.85**	**29.62**	**41.71**	**51.35**	**60.45**	**41.29**
TableLLM (w/ SFT)			22.30	30.77	21.92	27.17	38.84	39.00	30.00
RE-TableLLM (w/ RE-SFT)	Llama3.1	8B	28.71	38.36	30.46	42.14	61.03	62.92	43.94
Table-R1 (w/ RE-SFT & TARPO)			**38.5**	**40.21**	**38.51**	**42.78**	**63.51**	**64.30**	**47.97**
TableLLM (w/ SFT)			22.45	32.65	23.91	25.29	63.37	65.58	38.88
RE-TableLLM (w/ RE-SFT)	TeleChat2	7B	29.01	38.57	31.15	**41.86**	63.52	64.93	44.84
Table-R1 (w/ RE-SFT & TARPO)			**39.29**	**42.14**	**38.95**	41.60	**65.27**	**66.12**	**48.90**
TableLLM (w/ SFT)			20.08	30.02	22.11	27.55	55.50	61.46	36.12
RE-TableLLM (w/ RE-SFT)	Qwen2.5	3B	22.38	32.60	25.57	41.64	51.57	61.55	39.22
Table-R1 (w/ RE-SFT & TARPO)			**35.71**	**39.54**	**36.78**	**41.93**	**56.51**	**63.98**	**45.74**
TableLLM (w/ SFT)			22.42	35.88	27.58	29.15	73.02	71.14	43.20
RE-TableLLM (w/ RE-SFT)	Qwen3	8B	31.53	44.67	34.70	44.36	69.06	69.57	48.98
Table-R1 (w/ RE-SFT & TARPO)			**44.54**	**49.30**	**47.21**	**44.54**	**73.87**	**72.50**	**55.33**

Table 2: The experimental results on three datasets. We compare Table-R1 LLMs with general-purpose LLMs, table-focused LLMs, and same-base LLMs. TableLLM-RE indicates the model built on TableLLM with only RE-SFT training, while Table-R1 represents our full framework, including both RE-SFT and TARPO. The experimental results demonstrate the effectiveness of RE-SFT and TARPO and and shows that Table-R1 achieves better performance with different base models. Bold numbers indicate the best result within each experimental group.

tematically evaluates RE-SFT and TARPO's contributions. Results show RE-SFT delivers overall performance gains across three benchmarks, especially in TableBench PoT data, with minor declines observed on cross-domain test sets (WikiTQ and WikiSQL). Across seven models, RE-SFT achieves an average PoT score increase of 19.90 and an overall increase of 8.45. TARPO integration further enhances performance. As shown in Table 2, Table-R1 achieves significant improvement on DP, TCoT, and SCoT data of TableBench, WikiTQ and WikiSQL datasets, achieving an average gain of 4.52 in overall score than RE-SFT alone. In conclusion, Table-R1 gets the top-tier performance on different benchmarks by a large margin. Especially, with the same base model Qwen2.5, 3B-sized Table-R1 surpasses 7B-sized Qwen2.5-Coder, Qwen2.5-Instruct and TableGPT2 overall. On Qwen3-8B, Table-R1 exceeds GPT-4o by 6.16 overall, underperforming only by 0.76 on TCoT.

4.2 Ablation Study

We use Qwen3-8B as the base model for ablation study. Table 3 presents the experimental results.

Compared to the baseline TableLLM (Wu et al., 2024), RE-SFT significantly enhances the model's performance on TableBench, especially with a 15.21 gain in PoT data. Yet, it causes performance drops of 3.96 and 1.57 on the other two out-of-domain test sets WikiTQ and WikiSQL. Integrating table regions into CoT also increases average response tokens across the three test sets by 129%,

Model	TableBench					WikiTQ		WikiSQL	
	DP	TCoT	SCoT	PoT	Avg tokens	Acc	Avg tokens	Acc	Avg tokens
w/ SFT (TableLLM)	22.42	35.88	27.58	29.15	**581**	73.02	**313**	71.14	**201**
w/ RE-SFT	31.53 ($\uparrow$ 9.81)	44.67 ($\uparrow$ 8.79)	34.70 ($\uparrow$ 7.12)	44.36 ($\uparrow$ 15.21)	1,330 ($\uparrow$ 129%)	69.06 ($\downarrow$ 3.96)	1,028 ($\uparrow$ 228%)	69.57 ($\downarrow$ 1.57)	296 ($\uparrow$ 47.3%)
w/ RE-SFT & GRPO	42.41 ($\uparrow$ 10.88)	48.64 ($\uparrow$ 3.97)	45.62 ($\uparrow$ 10.92)	42.44 ($\downarrow$ 1.92)	1,549 ($\uparrow$ 16.5%)	73.15 ($\uparrow$ 4.09)	996 ($\downarrow$ 0.03%)	70.24 ($\uparrow$ 0.67)	1,399 ($\uparrow$ 373%)
w/ RE-SFT & TARPO (w/o $\mathcal{P}$)	44.43 ($\uparrow$ 2.02)	49.15 ($\uparrow$ 0.51)	46.88 ($\uparrow$ 1.26)	44.20 ($\uparrow$ 1.76)	630 ($\downarrow$ 59.3%)	73.09 ($\downarrow$ 0.06)	416 ($\downarrow$ 58.2%)	70.83 ($\uparrow$ 0.59)	276 ($\downarrow$ 80.3%)
w/ RE-SFT & TARPO (w/ $\mathcal{P}$)	**44.54** ($\uparrow$ 0.11)	**49.30** ($\uparrow$ 0.15)	**47.21** ($\uparrow$ 0.33)	**44.54** ($\uparrow$ 0.34)	612 ($\downarrow$ 0.03%)	**73.87** ($\uparrow$ 0.78)	397 ($\downarrow$ 0.05%)	**72.50** ($\uparrow$ 1.67)	251 ($\downarrow$ 0.09%)

Table 3: The ablation study of Table-R1 based on Qwen3-8B. Blue indicates better performance (higher scores and lower average tokens), while red shows worse performance (lower scores and higher average tokens).

228%, and 47.3% respectively. We further employ GRPO algorithm for reinforcement learning. Experiment results show marked improvement in Tablebench DP, TCoT, SCoT, and WikiTQ performance, while PoT performance slightly decreases. GRPO also causes a rise in response reasoning tokens, with TableBench and WikiSQL seeing average increases of 16.5% and 373%. Then, we substitute GRPO with the TARPO algorithm (without penalty $\mathcal{P}$). The results demonstrate slight performance enhancements on TableBench and WikiSQL, while experiencing a minor decline on WikiTQ. Notably, TARPO achieve a significant reduction in average response tokens compared to GRPO, with decreases of 59.3%, 58.2%, and 80.3% across three datasets, which suggests that TARPO can guide a more concise and direct CoT process. Furthermore, we add the penalty $\mathcal{P}$ in TARPO to achieve consistency preference. The results show a slight improvement on TableBench and more noticeable enhancements on WikiTQ and WikiSQL by 0.78% and 1.67%. This indicates that the penalty $\mathcal{P}$ effectively constrain the optimization consistency between table region and answer in training. It strengthens the rationality of the reasoning process by preventing the LLM from learning low-quality reasoning strategies that generate correct answers accidentally. Consequently, it improves the generalization of LLM on out-of-domain datasets.

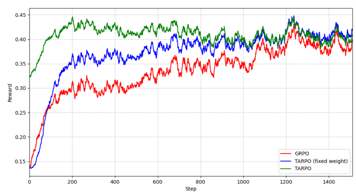
In conclusion, the whole Table-R1 framework achieves optimal performance across all three datasets. Meanwhile, when incorporating table regions, TARPO increases the average response tokens by 5.3%, 26.8%, and 24.9% respectively on the three datasets compared to TableLLM, but reduces them by an average of 60.5%, 60.1%, and 82.1% compared to the GRPO, which effectively addresses the excessive reasoning length caused by RE-SFT and GRPO.

4.3 Stastics Analysis

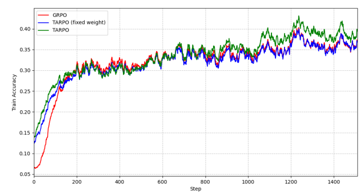
We compare the data stastics of GRPO, fixed-weight TARPO ($\alpha_\tau = 0.15$), and TARPO during reinforcement training. Figure 3 shows the mean reward, train accuracy, valid accuracy, and global token length as step increases. To better observe the overall trends, we apply EMA (Exponential Moving Average) to the three metrics except validation accuracy, with a decay factor of 0.05. It can be seen that TARPO's high initial α_τ value keeps its mean reward high in the early training phase. However, as α_τ decreases with step, the final reward of TARPO ends up higher than GRPO but lower than fixed-weight TARPO. This is because fixed-weight TARPO continues to benefit from the reward of the table region part due to the retained α_τ value at the end of training. Figure 3 (b) and (c) illustrate the accuracy of the current batch and the validation set during training, respectively. TARPO demonstrates better performance compared to the other two methods, indicating both its advantage over GRPO and the effectiveness of dynamic weights. Figure 3 (d) shows the global token length per step. TARPO shows a superior ability in reducing output token length compared to GRPO and also exhibits a more significant reduction than fixed-weight TARPO. Figure 3 (e) and (f) demonstrate that the col & row-based table structure surpasses the cell-based structure in terms of performance and token conservation.

5 Related Work

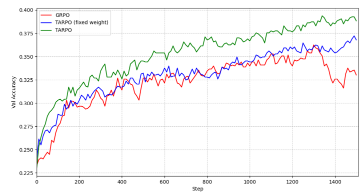
Table Understanding. The evolution of Table Question Answering (Table QA) research (Mueller et al., 2019; Yang et al., 2020b; Jin et al., 2022; Yang et al., 2025a, 2024c; Hui et al., 2024; Lin



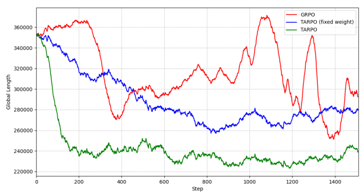
(a) Mean Reward



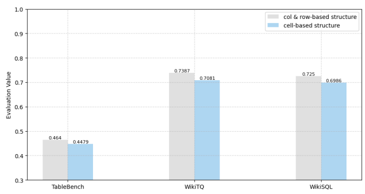
(b) Train Accuracy



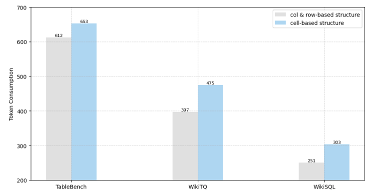
(c) Val Accuracy



(d) Global Length (bz=32)



(e) Evaluation Value



(f) Token Consumption

Figure 3: (a)-(d) Reinforcement learning training data statistics for TableBench on Qwen3-8B. (e)-(f) The impact of col & row-based table structure and cell-based structure on evaluation value and token consumption.

et al., 2024; Wu and Feng, 2024; Wu et al., 2025a; Li et al., 2024b; Wu et al., 2025b; Ye et al., 2025; Guo et al., 2026, 2025b; Hong et al., 2025, 2024; Yuan et al., 2025; Zhang et al., 2024a) has been propelled by the creation of sophisticated evaluation resources that facilitate semantic parsing capabilities (Huang et al., 2024; Yang et al., 2020a; Li et al., 2023, 2024a; Nakamura et al., 2022; Anirudh S Sundar, 2023; Bai et al., 2023; Yang et al., 2024a,b, 2025g; Chai et al., 2024; Zhang et al., 2024b). Foundational works including WTQ (Pasupat and Liang, 2015b), SQA (Iyyer et al., 2017), and TabFact (Chen et al., 2019) established initial evaluation paradigms through Wikipedia-derived HTML table QA pairs. While these resources provide structured testing grounds, their answer formulations predominantly depend on direct cell retrieval, limiting their capacity to emulate the multi-faceted information needs observed in practical applications. Recent initiatives address this limitation through innovative dataset designs. ToTTo (Parikh et al., 2020) and FeTaQA (Nan et al., 2022) pioneer open-domain QA frameworks requiring generative responses that synthesize implicit table relationships. Alternative approaches employ structured supervision signals, exemplified by WikiSQL (Zhong et al., 2017) and Spider (Yu et al., 2018), which utilize logical expressions to cultivate systematic reasoning skills.

RL for LLMs. Driven by the widespread expansion of LLM applications (Chang et al., 2024, 2026; Dong et al., 2025, 2026; Jiang et al., 2026; Yang et al., 2026b,c, 2025e,d,f,c, 2026a), recent advancements in reinforcement learning (RL) frameworks have demonstrated significant improvements in augmenting the inferential capacities of LLMs, exemplified by architectures like DeepSeek-R1 (Guo et al., 2025a) and OpenAI-o1 (Jaech et al., 2024). Driven by these breakthroughs, a burgeoning body of research has increasingly deployed reinforcement learning paradigms across diverse specialized domains, facilitating autonomous knowledge discovery and systematic exploration within complex task environments (Wu et al., 2026; Zhao et al., 2026; Zhou et al., 2026; Hao et al., 2025; Liu et al., 2025b; Bai et al., 2026; Chang et al., 2025). Through cyclical switching between exploratory response generation and strategic knowledge utilization, these frameworks progressively refine output granularity while achieving sustained performance gains. State-of-the-art implementations predominantly leverage policy optimization techniques, including Proximal Policy Optimization (PPO) (Schulman et al., 2017) and the parameter-efficient Group Relative Policy Optimization (GRPO) (Shao et al., 2024). The latter methodology eliminates dependency on auxiliary value estimation networks, thereby achieving enhanced computational economy.

6 Conclusion

In this paper, we introduce Table-R1, a novel region-based reinforcement learning method for table understanding. Through RE-SFT and TARPO training, Table-R1 effectively incorporates the extraction of the minimum table region into the reasoning CoT, thereby enhancing the guidance of answer generation. The experimental results show that Table-R1 delivers an average improvement of 14.36 points across multiple base models on three benchmarks. Moreover, TARPO reduces response token consumption by an average of 67.5% compared to GRPO. In future work, we hope Table-R1 will inspire more research on rule-based reinforcement learning methods for different domain-specific tasks. We also aim to encourage the exploration of signals beyond final answers in the CoT process that can be used for reward shaping to enhance the quality of reasoning process.

Limitations

Our experiments are conducted exclusively on models with 3B to 8B parameters and are limited to maximum input and output lengths of 8192 and 2048 tokens, respectively. The results may vary for larger models or longer token sequences. In addition, for simplicity, we only inserted the table region midway into CoT without refining the CoT process, the results may not be optimal. Our current study focuses on established academic datasets and has not yet been validated on real-world complex tables in industrial settings or multi-table QA scenarios.

Ethics Statement

In this work, all of the datasets, models, code and related documents are not associated with any ethical concerns.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 62276017, 62406033, U1636211, 61672081), the State Key Laboratory of Complex & Critical Software Environment (Grant Nos. SKLCCSE-2024ZX-18, SKLCCSE-2025ZX-26), and the Fundamental Research Funds for the Central Universities (Grant No. GW2025-19).

References

01.AI. 2024. Meet yi-coder: A small but mighty llm for code.

Larry Heck Anirudh S Sundar. 2023. ctbls: Augmenting large language models with conversational tables. *arXiv preprint arXiv:2303.12024*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Sikai Bai, Haoxi Li, Jie Zhang, Yongjiang Liu, and Song Guo. 2026. Ttvs: Boosting self-exploring reinforcement learning via test-time variational synthesis.

Linzheng Chai, Jian Yang, Tao Sun, Hongcheng Guo, Jiaheng Liu, Bing Wang, Xinnian Liang, Jiaqi Bai, Tongliang Li, Qiyao Peng, and Zhoujun Li. 2024. xcot: Cross-lingual instruction tuning for cross-lingual chain-of-thought reasoning. *arXiv preprint arXiv:2401.07037*, abs/2401.07037.

Yupeng Chang, Yi Chang, and Yuan Wu. 2026. BA-loRA: Bias-alleviating low-rank adaptation to mitigate catastrophic inheritance in large language models. In *The Fourteenth International Conference on Learning Representations*.

Yupeng Chang, Chenlu Guo, Yi Chang, and Yuan Wu. 2025. Lora-mgpo: Mitigating double descent in low-rank adaptation via momentum-guided perturbation optimization. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 648–659.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and 1 others. 2024. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *CoRR*, abs/2107.03374.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, abs/2211.12588.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2019. Tabfact: A large-scale dataset for table-based fact verification. *CoRR*.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Binding language models in symbolic languages. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Daya Guo, and 1 others. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *CoRR*, abs/2405.04434.

Haonan Dong, Kehan Jiang, Haoran Ye, Wenhao Zhu, Zhaolu Kang, and Guojie Song. 2026. Neureasoner: Towards explainable, controllable, and unified reasoning via mixture-of-neurons. *Preprint*, arXiv:2604.02972.

Haonan Dong, Wenhao Zhu, Guojie Song, and Liang Wang. 2025. AuroRA: Breaking low-rank bottleneck of loRA with nonlinear mapping. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y Wu, YK Li, and 1 others. 2024. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*.

Yu Guo, Dong Jin, Shenghao Ye, Shuangwu Chen, Jianyang Jianyang, and Xiaobin Tan. 2025b. Sqlforge: Synthesizing reliable and diverse data to enhance text-to-sql reasoning in llms. In *Findings of the Association for Computational Linguistics: ACL 2025*.

Yu Guo, Shenghao Ye, Shuangwu Chen, Zijian Wen, Tao Zhang, Qirui Bai, Dong Jin, Yunpeng Hou, Huasen He, Jian Yang, and 1 others. 2026. Rethinking table pruning in tableqa: From sequential revisions to gold trajectory-supervised parallel search. *arXiv preprint arXiv:2601.03851*.

Zhezheng Hao, Hong Wang, Haoyang Liu, Jian Luo, Jiarui Yu, Hande Dong, Qiang Lin, Can Wang, and Jiawei Chen. 2025. Rethinking entropy interventions in rlvr: An entropy change perspective. *arXiv preprint arXiv:2510.10150*.

Zijin Hong, Zheng Yuan, Hao Chen, Qinggang Zhang, Feiran Huang, and Xiao Huang. 2024. Knowledge-to-sql: Enhancing sql generation with data expert llm. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10997–11008.

Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. 2025. Next-generation database interfaces: A survey of llm-based text-to-sql. *IEEE Transactions on Knowledge and Data Engineering*.

Shulin Huang, Shirong Ma, Yinghui Li, Mengzuo Huang, Wuhe Zou, Weidong Zhang, and Haitao Zheng. 2024. Lateval: An interactive llms evaluation benchmark with incomplete information from lateral thinking puzzles. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation, LREC/COLING 2024, 20-25 May, 2024, Torino, Italy*, pages 10186–10197. ELRA and ICCL.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, and 1 others. 2024. Qwen2.5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *ACL 2017*, pages 1821–1831.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Kehan Jiang, Haonan Dong, Zhaolu Kang, Zhengzhou Zhu, and Guojie Song. 2026. Foe: Forest of errors makes the first solution the best in large reasoning models. *Preprint*, arXiv:2604.02967.

Nengzheng Jin, Joanna Siebert, Dongfang Li, and Qingcai Chen. 2022. A survey on table question answering: recent advances. In *China Conference on Knowledge Graph and Semantic Computing*, pages 174–186. Springer.

Yinghui Li, Zishan Xu, Shaoshen Chen, Haojing Huang, Yangning Li, Yong Jiang, Zhongli Li, Qingyu Zhou, Hai-Tao Zheng, and Ying Shen. 2023. Towards real-world writing assistance: A chinese character checking benchmark with faked and misspelled characters. *CoRR*, abs/2311.11268.

Yinghui Li, Qingyu Zhou, Yuanzhen Luo, Shirong Ma, Yangning Li, Hai-Tao Zheng, Xuming Hu, and Philip S. Yu. 2024a. When llms meet cunning questions: A fallacy understanding benchmark for large language models. *CoRR*, abs/2402.11100.

Zhongqiu Li, Zhenhe Wu, Mengxiang Li, Zhongjiang He, Ruiyu Fang, Jie Zhang, Yu Zhao, Yongxiang Li, Zhoujun Li, and Shuangyong Song. 2024b. Scalable database-driven kgs can help text-to-sql. In *Proceedings of the ISWC 2024 Posters, Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 23nd International Semantic Web Conference (ISWC 2024), Hanover, Maryland, USA, November 11-15, 2024*, CEUR Workshop Proceedings. CEUR-WS.org.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, Barcelona, Spain. Association for Computational Linguistics.

Yupian Lin, Tong Ruan, Jingping Liu, and Haofen Wang. 2024. A survey on neural data-to-text generation. *IEEE Trans. Knowl. Data Eng.*, 36(4):1431–1449.

Xinzhang Liu, Chao Wang, Zhihao Yang, Zhuo Jiang, Xudong Zhao, Haoran Wang, Lei Li, Dongdong He, Luobin Liu, Kaizhe Yuan, Han Gao, Zihan Wang, Yitong Yao, Sishi Xiong, Wenmin Deng, Haowei He, Kaidong Yu, Yu Zhao, Ruiyu Fang, and 35 others. 2025a. Training report of telechat3-moe. *CoRR*, abs/2512.24157.

Zheng Liu, Mengjie Liu, Siwei Wen, Mengzhang Cai, Bin Cui, Conghui He, and Wentao Zhang. 2025b. From uniform to heterogeneous: Tailoring policy optimization to every token's nature. *arXiv preprint arXiv:2509.16591*.

Thomas Mueller, Francesco Piccinno, Peter Shaw, Massimo Nicosia, and Yasemin Altun. 2019. Answering conversational questions on structured data without logical forms. In *EMNLP-IJCNLP 2019*, pages 5902–5910.

Md Mahadi Hasan Nahid and Davood Rafiei. 2024. Tabsqlify: Enhancing reasoning capabilities of llms through table decomposition. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 5725–5737. Association for Computational Linguistics.

Kai Nakamura, Sharon Levy, Yi-Lin Tuan, Wenhu Chen, and William Yang Wang. 2022. Hybridialogue: An information-seeking dialogue dataset grounded on tabular and textual data. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 481–492. Association for Computational Linguistics.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, and 1 others. 2022. Fetaqa: Free-form table question answering. *TACL 2022*, 10:35–49.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

OpenAI. 2025. Introducing gpt-4.5.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. Totto: A controlled table-to-text generation dataset. In *EMNLP 2020*, pages 1173–1186.

Panupong Pasupat and Percy Liang. 2015a. 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 of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*.

Panupong Pasupat and Percy Liang. 2015b. Compositional semantic parsing on semi-structured tables. In *ACL 2015*, pages 1470–1480.

Joseph Redmon and Ali Farhadi. 2017. YOLO9000: better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6517–6525. IEEE Computer Society.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, and 1 others. 2023. Code llama: Open foundation models for code. *CoRR*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. *URL https://arxiv.org/abs/2402.03300*.

Aofeng Su, Aowen Wang, Chao Ye, Chen Zhou, Ga Zhang, Guangcheng Zhu, Haobo Wang, Haokai Xu, Hao Chen, Haoze Li, and 1 others. 2024. Tablegpt2: A large multimodal model with tabular data integration. *arXiv preprint arXiv:2411.02059*.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2024. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 645–654.

Zihan Wang, XinZhang Liu, Shixuan Liu, Yitong Yao, Yunyao Huang, Mengxiang Li, Zhongjiang He, Yongxian Li, Luwen Pu, Huinan Xu, Chao Wang, and Shuangyong Song. 2024a. TeleChat: An open-source billingual large language model. In *Proceedings of the 10th SIGHAN Workshop on Chinese Language Processing (SIGHAN-10)*.

Zihan Wang, Xinzhang Liu, Shixuan Liu, Yitong Yao, Yuyao Huang, Zhongjiang He, Xuelong Li, Yongxiang Li, Zhonghao Che, Zhaoxi Zhang, Yan Wang, Xin Wang, Luwen Pu, Huihan Xu, Ruiyu Fang, Yu Zhao, Jie Zhang, Xiaomeng Huang, Zhilong Lu, and 17 others. 2024b. Telechat technical report. *CoRR*, abs/2401.03804.

Zihan Wang, Xinzhang Liu, Yitong Yao, Chao Wang, Yu Zhao, Zhihao Yang, Wenmin Deng, Kaipeng Jia, Jiaxin Peng, Yuyao Huang, Sishi Xiong, Zhuo Jiang, Kaidong Yu, Xiaohui Hu, Fubei Yao, Ruiyu Fang, Zhuoru Jiang, Ruiting Song, Qiyi Xie, and 19 others. 2025. Technical report of telechat2, telechat2.5 and T1. *CoRR*, abs/2507.18013.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2024c. Chain-of-table: Evolving tables in the reasoning chain for table understanding. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and 1 others. 2024d. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *arXiv preprint arXiv:2401.04398*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Fei Wu, Zhenrong Zhang, Qikai Chang, Jianshu Zhang, Quan Liu, and Jun Du. 2026. Step potential advantage estimation: Harnessing intermediate confidence and correctness for efficient mathematical reasoning. *arXiv preprint arXiv:2601.03823*.

Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xinrun Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, and 1 others. 2024. Tablebench: A comprehensive and complex benchmark for table question answering. *arXiv preprint arXiv:2408.09174*.

Zhenhe Wu, Zhongqiu Li, Mengxiang Li, Jie Zhang, Zhongjiang He, Jian Yang, Yu Zhao, Ruiyu Fang, Yongxiang Li, Zhoujun Li, and Shuangyong Song. 2025a. MR-SQL: multi-level retrieval enhances inference for LLM in text-to-sql. In *Database Systems for Advanced Applications - 30th International Conference, DASFAA 2025, Singapore, Singapore, May 26-29, 2025, Proceedings, Part II*, Lecture Notes in Computer Science, pages 403–413. Springer.

Zhenhe Wu, Zhongqiu Li, Jie Zhang, Zhongjiang He, Jian Yang, Yu Zhao, Ruiyu Fang, Bing Wang, Hongyan Xie, Shuangyong Song, and Zhoujun Li. 2025b. UCS-SQL: uniting content and structure for enhanced semantic bridging in text-to-sql. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, Findings of ACL, pages 8156–8168. Association for Computational Linguistics.

Zirui Wu and Yansong Feng. 2024. ProTrix: Building models for planning and reasoning over tables with sentence context. In *Findings of the Association for Computational Linguistics: EMNLP 2024*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, and 1 others. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2025b. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Jian Yang, Xianglong Liu, Weifeng Lv, Ken Deng, Shawn Guo, Lin Jing, Yizhi Li, Shark Liu, Xianzhen Luo, Yuyu Luo, and 1 others. 2025c. From code foundation models to agents and applications: A comprehensive survey and practical guide to code intelligence. *arXiv preprint arXiv:2511.18538*.

Jian Yang, Shuming Ma, Dongdong Zhang, Zhoujun Li, and Ming Zhou. 2020a. Improving neural machine translation with soft template prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5979–5989. Association for Computational Linguistics.

Jian Yang, Shuming Ma, Dongdong Zhang, Shuangzhi Wu, Zhoujun Li, and Ming Zhou. 2020b. Alternating language modeling for cross-lingual pre-training. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9386–9393. AAAI Press.

Jian Yang, Jiaxi Yang, Ke Jin, Yibo Miao, Lei Zhang, Liqun Yang, Zeyu Cui, Yichang Zhang, Binyuan Hui, and Junyang Lin. 2024b. Evaluating and aligning codellms on human preference. *arXiv preprint arXiv:2412.05210*.

Jian Yang, Jiaxi Yang, Wei Zhang, Ke Jin, Yibo Miao, Lei Zhang, Liqun Yang, Zeyu Cui, Yichang Zhang, Zhoujun Li, Binyuan Hui, and Junyang Lin. 2025d. Codearena: Evaluating and aligning codellms on human preference. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language*

Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025, pages 9672–9683. Association for Computational Linguistics.

Jian Yang, Jiajun Zhang, Jiaxi Yang, Ke Jin, Lei Zhang, Qiyao Peng, Ken Deng, Yibo Miao, Tianyu Liu, Zeyu Cui, and 1 others. 2024c. Execrepobench: Multi-level executable code completion evaluation. *arXiv preprint arXiv:2412.11990*.

Jian Yang, Wei Zhang, Shawn Guo, Zhengmao Ye, Lin Jing, Shark Liu, Yizhi Li, Jiajun Wu, Cening Liu, X Ma, and 1 others. 2026a. Iquest-coder-v1 technical report. *arXiv preprint arXiv:2603.16733*.

Jian Yang, Wei Zhang, Yizhi Li, Shawn Guo, Haowen Wang, Aishan Liu, Ge Zhang, Zili Wang, Zhoujun Li, Xianglong Liu, and 1 others. 2025e. Codesimpleqa: Scaling factuality in code large language models. *arXiv preprint arXiv:2512.19424*.

Jian Yang, Wei Zhang, Yibo Miao, Shanghaoran Quan, Zhenhe Wu, Qiyao Peng, Liqun Yang, Tianyu Liu, Zeyu Cui, Binyuan Hui, and Junyang Lin. 2025f. Qwen2.5-xcoder: Multi-agent collaboration for multilingual code instruction tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 13121–13131. Association for Computational Linguistics.

Jian Yang, Wei Zhang, Jiajun Wu, Junhang Cheng, Shawn Guo, Haowen Wang, Weicheng Gu, Yaxin Du, Joseph Li, Fanglin Xu, and 1 others. 2026b. Incoder-32b: Code foundation model for industrial scenarios. *arXiv preprint arXiv:2603.16790*.

Jian Yang, Wei Zhang, Jiajun Wu, Junhang Cheng, Tuney Zheng, Fanglin Xu, Weicheng Gu, Lin Jing, Yaxin Du, Joseph Li, and 1 others. 2026c. Incoder-32b-thinking: Industrial code world model for thinking. *arXiv preprint arXiv:2604.03144*.

Jian Yang, Wei Zhang, Jiaxi Yang, Yibo Miao, Shanghaoran Quan, Zhenhe Wu, Qiyao Peng, Liqun Yang, Tianyu Liu, Zeyu Cui, and 1 others. 2025g. Multi-agent collaboration for multilingual code instruction tuning. *arXiv preprint arXiv:2502.07487*.

Shenghao Ye, Yu Guo, Dong Jin, Yikai Shen, Yunpeng Hou, Shuangwu Chen, Jian Yang, and Xiaofeng Jiang. 2025. When tableqa meets noise: A dual denoising framework for complex questions and large-scale tables. *arXiv preprint arXiv:2509.17680*.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, pages 174–184. ACM.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, and 1 others. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *EMNLP 2018*, pages 3911–3921.

Zheng Yuan, Hao Chen, Zijin Hong, Qinggang Zhang, Feiran Huang, Qing Li, and Xiao Huang. 2025. Knapsack optimization-based schema linking for llm-based text-to-sql generation. *arXiv preprint arXiv:2502.12911*.

Qinggang Zhang, Hao Chen, Junnan Dong, Shengyuan Chen, Feiran Huang, and Xiao Huang. 2024a. Structure guided large language model for sql generation. *arXiv preprint arXiv:2402.13284*.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2024b. Tablellama: Towards open large generalist models for tables. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 6024–6044. Association for Computational Linguistics.

Xiaokang Zhang, Jing Zhang, Zeyao Ma, Yang Li, Bohan Zhang, Guanlin Li, Zijun Yao, Kangli Xu, Jinchang Zhou, Daniel Zhang-Li, Jifan Yu, Shu Zhao, Juanzi Li, and Jie Tang. 2024c. Tablellm: Enabling tabular data manipulation by llms in real office usage scenarios. *CoRR*, abs/2403.19318.

Ziqi Zhao, Zhaochun Ren, Jiahong Zou, Liu Yang, Zhiwei Xu, Xuri Ge, Zhumin Chen, Xinyu Ma, Daiting Shi, Shuaiqiang Wang, and 1 others. 2026. Reinforced efficient reasoning via semantically diverse exploration. *arXiv preprint arXiv:2601.05053*.

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

Yixiao Zhou, Yang Li, Dongzhou Cheng, Hehe Fan, and Yu Cheng. 2026. Look inward to explore outward: Learning temperature policy from llm internal states via hierarchical rl. *arXiv preprint arXiv:2602.13035*.

A How RE-SFT enhances the performance of PoT

Method	Base Model	Size	FC	NR	DA	VIZ	Overall
SFT	CodeQwen	7B	11.46	18.64	14.28	36.00	15.14
RE-SFT			64.58	55.42	27.57	40.00	42.52
SFT	DS-Coder	7B	10.42	23.17	18.58	30.00	18.72
RE-SFT			66.67	55.16	25.01	36.00	41.65
SFT	Llama3	8B	12.50	16.88	15.02	28.00	14.75
RE-SFT			61.46	53.40	26.73	28.00	40.95
SFT	Llama3.1	8B	27.08	38.29	18.16	44.00	27.14
RE-SFT			60.42	55.92	27.20	32.00	42.15
SFT	Qwen2	7B	9.38	13.35	15.10	26.00	12.86
RE-SFT			61.46	52.14	18.08	22.00	41.07
SFT	Qwen2.5	3B	38.54	30.98	24.48	34.00	27.55
RE-SFT			59.38	52.90	29.66	32.00	41.64
SFT	Qwen3	8B	42.71	33.50	24.52	42.00	29.15
RE-SFT			67.71	56.93	29.70	38.00	44.36

Table 4: Detailed results on PoT of TableBench. There are four types of questions: Fact Checking, Numerical Reasoning, Data Analysis, and Visualization.

In the main experiment results, RE-SFT significantly enhances the performance of PoT. Consequently, we conduct a detailed classification of questions to investigate the specific improvements in each category. As shown in Table 4, RE-SFT leads to substantial improvements in Fact Checking and Numerical Reasoning questions. It also results in partial enhancements in Data Analysis. However, it exerts no significant influence on Visualization.

We've conducted a case study on PoT data to investigate how model reasoning evolves after SFT and RE-SFT. Figure 4 illustrates that RE-SFT model initially identifies a narrow table region, then generates code based on it, streamlining the code generation process. This approach reduces errors from ambiguous or incorrect reasoning compared to the SFT model.

B LLM Prompt Examples

Figure 5, 6, 7, 8 present the detailed prompts used to elicit the four distinct reasoning methods, while Figure 9 shows the instructions provided to DeepSeek-R1 for inserting table regions within the chain-of-thought.

Question: What was the ranking of guangdong in 2008?

Table:

Year	2010	2009	2008	2005	2000
shanghai	1	2	1	1	1
beijing	2	1	2	2	2
tianjin	3	3	3	3	3
jiangsu	4	4	5	6	6
zhejiang	5	4	4	4	4
inner mongolia	6	6	7	10	15
guangdong	7	7	6	5	5
liaoning	8	9	9	8	8
shandong	9	8	8	7	9
fujian	10	10	10	9	7
jilin	11	11	11	13	13
hebei	12	12	12	11	1
hubei	13	14	16	17	16
chongqing	14	13	15	16	17
shaanxi	15	16	18	20	23
heilongjiang	16	15	13	12	10
ningxia	17	17	19	22	21
shanxi	18	18	14	15	18
xinjiang	19	21	17	14	12
hunan	20	20	22	21	20
henan	21	19	20	18	19
qinghai	22	22	21	23	22
hainan	23	23	23	19	14
jiangxi	24	25	24	24	25
sichuan	25	24	25	25	24
anhui	26	26	27	27	26
guangxi	27	27	26	28	28
taibet	28	28	28	26	29
gansu	29	30	30	30	30
yunnan	30	29	29	29	27
guizhou	31	31	31	31	31

Gold Answer: 6

SFT Response:

~~~
To determine the ranking of Guangdong in 2008, we need to follow these steps:

1. Load the data from the CSV file.
2. Extract the row corresponding to Guangdong.
3. Identify the value in the '2008' column for Guangdong.
4. Determine the ranking based on the extracted value.

Let's write the Python code to achieve this:

```python
import pandas as pd

# Load the data
df = pd.read_csv('table.csv')

# Extract the row for Guangdong
guangdong_row = df[df['year'] == 'guangdong']

# Get the value in the '2008' column
ranking_2008 = guangdong_row['2008'].values[0]

# Since the ranking is the position in the sorted list, we need to find its rank
rank = (len(df) - len(guangdong_row)) + 1

print(f\"Final Answer: {rank}\")
```

This code will load the data, extract the relevant row, and determine the ranking of Guangdong in 2008. The final answer will be printed as the rank.
~~~

Execution Result: 31 ✗

RE-SFT Response:

~~~
To determine the ranking of Guangdong in 2008, we need to follow these steps:
1. Load the data from the CSV file.
2. Filter the data to get the row corresponding to Guangdong.

Up to this point, we can identify the table area needed to answer the question:
<gold_area>{\"columns\": [\"year\", \"2010\", \"2009\", \"2008\", \"2005\", \"2000\"], \"rows\": [6], \"data\": [[\"guangdong\", 7, 7, 6, 5, 5]]}</gold_area>

3. Extract the value for the year 2008.
4. Determine the ranking based on the value.

Let's write the Python code to perform this analysis.

```python
import pandas as pd

# Load the data\ndf = pd.read_csv('table.csv')

# Filter the data to get the row for Guangdong
guangdong_data = df[df['year'] == 'guangdong']

# Extract the value for the year 2008
ranking_2008 = guangdong_data['2008'].values[0]

# Determine the ranking
ranking = ranking_2008

print(f'Final Answer: {ranking}')\n```
~~~

Execution Result: 6 ✓

Figure 4: A case study for PoT, comparing SFT and RE-SFT.

You are a table analyst. Your task is to answer questions based on the table content.

The answer should follow the format below:
[Answer Format]
Final Answer: AnswerName1, AnswerName2...

Ensure the final answer format is the last output line and can only be in the "Final Answer: AnswerName1, AnswerName2..." form, no other form. Ensure the "AnswerName" is a number or entity name, as short as possible, without any explanation.

First, give the formatted table area result, then give the final answer to the question directly without any explanation.
Example Table Area Format:
<gold_area>{"columns": ["column1", "column2", "column3", "column4"], "rows": [0,3]}</gold_area>
~~Give the final answer to the question directly without any explanation.~~

Read the table below in JSON format:
[TABLE]
{"columns": ["season", "tropical lows", "tropical cyclones", "severe tropical cyclones", "strongest storm"], "data": [["1990 - 91", 10, 10, 7, "marian"], ["1991 - 92", 11, 10, 9, "jane - irna"], ["1992 - 93", 6, 3, 1, "oliver"], ["1993 - 94", 12, 11, 7, "theodore"], ["1994 - 95", 19, 9, 6, "chloe"], ["1995 - 96", 19, 14, 9, "olivia"], ["1996 - 97", 15, 14, 3, "pancho"], ["1997 - 98", 10, 9, 3, "tiffany"], ["1998 - 99", 21, 14, 9, "gwenda"], ["1999 - 00", 13, 12, 5, "john / paul"]]}

Let's get start!
Question: What is the average number of tropical cyclones per season?

Figure 5: Instruction for DP data in TableBench.

You are a table analyst. Your task is to answer questions based on the table content.

The answer should follow the format below:
[Answer Format]
Final Answer: AnswerName1, AnswerName2...

Ensure the final answer format is the last output line and can only be in the "Final Answer: AnswerName1, AnswerName2..." form, no other form. Ensure the "AnswerName" is a number or entity name, as short as possible, without any explanation.

In the reasoning process, once the table area is inferred, insert the formatted table area result here. The table area result must be enclosed in <gold_area> and </gold_area> tags and follow the specified format as the input table. Columns correspond to the column names, rows correspond to the row numbers (with row numbers starting from 0), and data corresponds to all the retained data.
Example Table Area Format:
<gold_area>{"columns": ["column1", "column2", "column3", "column4"], "rows": [0,3]}</gold_area>
~~Let's think step by step~~, and then give the final answer to the question.

Read the table below in JSON format:
[TABLE]
{"columns": ["season", "tropical lows", "tropical cyclones", "severe tropical cyclones", "strongest storm"], "data": [["1990 - 91", 10, 10, 7, "marian"], ["1991 - 92", 11, 10, 9, "jane - irna"], ["1992 - 93", 6, 3, 1, "oliver"], ["1993 - 94", 12, 11, 7, "theodore"], ["1994 - 95", 19, 9, 6, "chloe"], ["1995 - 96", 19, 14, 9, "olivia"], ["1996 - 97", 15, 14, 3, "pancho"], ["1997 - 98", 10, 9, 3, "tiffany"], ["1998 - 99", 21, 14, 9, "gwenda"], ["1999 - 00", 13, 12, 5, "john / paul"]]}

Let's get start!
Question: What is the average number of tropical cyclones per season?

Figure 6: Instruction for TCoT data in TableBench (along with WikiTQ and WikiSQL in our experiments).

You are a table analyst. Your task is to utilize the Python package 'pandas' to analyze the table and then answer questions.

[Guidelines]
You should act in following patterns step by step to analyze the table and then give the final answer:
[Action Patterns]
Thought: You should always think about what to do to interact with Python code base on Result
Action: the action can **ONLY** be single line python code
Result: Simulate the result of the execution of the python code in Action, analyse that result and decide whether to continue or not
(This thought/Action/Result can repeat N times)

The answer should follow the format below:
[Answer Format]
Final Answer: AnswerName1, AnswerName2...

Ensure the final answer format is the last output line and can only be in the "Final Answer: AnswerName1, AnswerName2..." form, no other form. Ensure the "AnswerName" is a number or entity name, as short as possible, without any explanation.

In the reasoning process, once the table area is inferred, insert the formatted table area result here. The table area result must be enclosed in <gold_area> and </gold_area> tags and follow the specified format as the input table. Columns correspond to the column names, rows correspond to the row numbers (with row numbers starting from 0), and data corresponds to all the retained data.
Example Table Area Format:
<gold_area>{"columns": ["column1", "column2", "column3", "column4"], "rows": [0,3]}</gold_area>
~~Let's think step by step~~, and then give the final answer to the question.
Ensure to have a concluding thought that verifies the table, observations and the question before giving the final answer.

Read the table below in JSON format:
[TABLE]
{"columns": ["season", "tropical lows", "tropical cyclones", "severe tropical cyclones", "strongest storm"], "data": [["1990 - 91", 10, 10, 7, "marian"], ["1991 - 92", 11, 10, 9, "jane - irna"], ["1992 - 93", 6, 3, 1, "oliver"], ["1993 - 94", 12, 11, 7, "theodore"], ["1994 - 95", 19, 9, 6, "chloe"], ["1995 - 96", 19, 14, 9, "olivia"], ["1996 - 97", 15, 14, 3, "pancho"], ["1997 - 98", 10, 9, 3, "tiffany"], ["1998 - 99", 21, 14, 9, "gwenda"], ["1999 - 00", 13, 12, 5, "john / paul"]]}

Let's get start!
Question: What is the average number of tropical cyclones per season?

Figure 7: Instruction for SCoT data in TableBench.

You are a data analyst proficient in Python. Your task is to write executable Python code to analyze the table and then answer questions.

[Guidelines]
You should act following requirements below:
1. based on the question, write out your analytical approach, and then write Python code according to this approach.
2. The code needs to be concise and easy to understand, and if necessary, add comments for clarification.
3. Code blocks need to strictly start with ```python and end with ```
4. Your analysis must be based entirely on the above data. If the user's question is not related to data analysis, please politely refuse.
5. You need to generate executable code. If there are results to be presented, please use the print function; if there are charts, please use the matplotlib library to draw them.
6. Ensure to load the table with command ```df = pd.read_csv('table.csv')```\n\n\nThe answer should follow the format below:
[Answer Format]
Final Answer: AnswerName1, AnswerName2...

Ensure the final answer format is the last output line and can only be in the "Final Answer: AnswerName1, AnswerName2..." form, no other form. Ensure the "AnswerName" is a number or entity name, as short as possible, without any explanation.

In the reasoning process, once the table area is inferred, insert the formatted table area result here. The table area result must be enclosed in <gold_area> and </gold_area> tags and follow the specified format as the input table. Columns correspond to the column names, rows correspond to the row numbers (with row numbers starting from 0), and data corresponds to all the retained data.
Example Table Area Format:
<gold_area>{"columns": ["column1", "column2", "column3", "column4"], "rows": [0,3]}</gold_area>
~~Let's think step by step~~, and then generate python code to analyze table and present the final answer to the question.

Read the table below in JSON format:
[TABLE]
{"columns": ["season", "tropical lows", "tropical cyclones", "severe tropical cyclones", "strongest storm"], "data": [["1990 - 91", 10, 10, 7, "marian"], ["1991 - 92", 11, 10, 9, "jane - irna"], ["1992 - 93", 6, 3, 1, "oliver"], ["1993 - 94", 12, 11, 7, "theodore"], ["1994 - 95", 19, 9, 6, "chloe"], ["1995 - 96", 19, 14, 9, "olivia"], ["1996 - 97", 15, 14, 3, "pancho"], ["1997 - 98", 10, 9, 3, "tiffany"], ["1998 - 99", 21, 14, 9, "gwenda"], ["1999 - 00", 13, 12, 5, "john / paul"]]}

Let's get start!
Question: What is the average number of tropical cyclones per season?

Figure 8: Instruction for PoT data in TableBench.

You are given a task containing a table, a question related to the table and a step-by-step reasoning process (Chain of Thought) to solve the question. Your task is to insert the formatted table area result at the appropriate point in the reasoning process. The table area result must be enclosed in <gold_area> and </gold_area> tags and follow the specified format as the input table. Columns correspond to the column names, rows correspond to the row numbers (with row numbers starting from 0), and data corresponds to all the retained data.

**Task:
1. Analyze the reasoning process.
2. Identify the rows and columns in the table that are indeed used to answer the question in the reasoning process.
3. In the reasoning process, once the table area is inferred, insert the formatted table area result here. After obtaining the gold area, be sure to verify its correctness using the table area actually employed in the reasoning process that follows.
4. Return the modified CoT process, strictly follow the output format: Starting with "Modified Reasoning:", and output the original Reasoning process only with the gold area at an appropriate position. Do not modify any other parts of the original Reasoning process and do not output any extra characters.

**Example Output Format:
Modified Reasoning:
To answer the question, we need to follow these steps:\n\n1. Load the data from the CSV file.\n2. Filter the data to include only the rows where the \"Chart\" column is \"Album Rock Tracks\".\n3. Further filter the data to include only the rows where the \"Position\" column is less than or equal to 10.\n4. Further filter the data to include only the rows where the \"Year\" column is \"1986\".\n\nUp to this point, we can identify the table area needed to answer the question:\n<gold_area>{\"columns\": [\"Single\"], \"rows\": [1,3,6], \"data\": [[\"\\\\Dreams\\\\\\\"\\\"\"], [\"\\\\Love Walks In\\\\\\\"\\\"\"], [\"\\\\Why Can't This Be Love\\\\\\\"\\\"\"]]}</gold_area>\n\n5. Count the number of unique singles that meet these criteria.\n\nLet's write the Python code to perform this analysis.\n\n```python\nimport pandas as pd \ndf = pd.read_csv('table.csv')\n\n# Filter for Album Rock Tracks chart\nalbum_rock_tracks = df[df['Chart'] == 'Album Rock Tracks']\n\n# Filter for positions in the top 10\ntop_10_singles = album_rock_tracks[album_rock_tracks['Position'] <= 10]\n\n# Count the number of unique singles\nunique_top_10_singles = top_10_singles['Single'].nunique()\n\nanswer = unique_top_10_singles\nprint(f'Final Answer: {answer}')\n```

Here is an example:
Question:
How many singles by Van Halen reached the top 10 on the Album Rock Tracks chart in 1986?
Table:
{\"columns\": [\"Year\", \"Single\", \"Chart\", \"Position\"], \"data\": [[1986, \"\\\\Best of Both Worlds\\\\\\\"\\\"\", \"Album Rock Tracks\", 12], [1986, \"\\\\Dreams\\\\\\\"\\\"\", \"Album Rock Tracks\", 6], [1986, \"\\\\Dreams\\\\\\\"\\\"\", \"Billboard Hot 100\", 22], [1986, \"\\\\Love Walks In\\\\\\\"\\\"\", \"Album Rock Tracks\", 4], [1986, \"\\\\Love Walks In\\\\\\\"\\\"\", \"Billboard Hot 100\", 22], [1986, \"\\\\Summer Nights\\\\\\\"\\\"\", \"Album Rock Tracks\", 33], [1986, \"\\\\Why Can't This Be Love\\\\\\\"\\\"\", \"Album Rock Tracks\", 1], [1986, \"\\\\Why Can't This Be Love\\\\\\\"\\\"\", \"Billboard Hot 100\", 3]]}
Reasoning:
To answer the question, we need to follow these steps:\n\n1. Load the data from the CSV file.\n2. Filter the data to include only the rows where the \"Chart\" column is \"Album Rock Tracks\".\n3. Further filter the data to include only the rows where the \"Position\" column is less than or equal to 10.\n4. Further filter the data to include only the rows where the \"Year\" column is \"1986\".\n5. Count the number of unique singles that meet these criteria.\n\nLet's write the Python code to perform this analysis.\n\n```python\nimport pandas as pd \ndf = pd.read_csv('table.csv')\n\n# Filter for Album Rock Tracks chart\nalbum_rock_tracks = df[df['Chart'] == 'Album Rock Tracks']\n\n# Filter for positions in the top 10\ntop_10_singles = album_rock_tracks[album_rock_tracks['Position'] <= 10]\n\n# Count the number of unique singles\nunique_top_10_singles = top_10_singles['Single'].nunique()\n\nanswer = unique_top_10_singles\nprint(f'Final Answer: {answer}')\n```
Modified Reasoning:
To answer the question, we need to follow these steps:\n\n1. Load the data from the CSV file.\n2. Filter the data to include only the rows where the \"Chart\" column is \"Album Rock Tracks\".\n3. Further filter the data to include only the rows where the \"Position\" column is less than or equal to 10.\n4. Further filter the data to include only the rows where the \"Year\" column is \"1986\".\n\nUp to this point, we can identify the table area needed to answer the question:\n<gold_area>{\"columns\": [\"Single\"], \"rows\": [1,3,6], \"data\": [[\"\\\\Dreams\\\\\\\"\\\"\"], [\"\\\\Love Walks In\\\\\\\"\\\"\"], [\"\\\\Why Can't This Be Love\\\\\\\"\\\"\"]]}</gold_area>\n\n5. Count the number of unique singles that meet these criteria.\n\nLet's write the Python code to perform this analysis.\n\n```python\nimport pandas as pd \ndf = pd.read_csv('table.csv')\n\n# Filter for Album Rock Tracks chart\nalbum_rock_tracks = df[df['Chart'] == 'Album Rock Tracks']\n\n# Filter for positions in the top 10\ntop_10_singles = album_rock_tracks[album_rock_tracks['Position'] <= 10]\n\n# Count the number of unique singles\nunique_top_10_singles = top_10_singles['Single'].nunique()\n\nanswer = unique_top_10_singles\nprint(f'Final Answer: {answer}')\n```
==============================
Question:
{question}
Table:
{table}
Reasoning:
{response}

Figure 9: Instructions for DeepSeek-R1 inserting Table Regions in CoT.