Rethinking Stepwise Model Routing: A Cost-Efficient Table Reasoning Perspective

Shenghao Ye[1*], **Yuxiang Wang**[2*], **Yu Guo**[1*], **Tao Zhang**[1], **Shuangwu Chen**[1†], **Zijian Wen**[1], **Qihui Zhu**[1], **Mengjie Zhang**[1], **Dong Jin**[4], **Wang Luo**[3], **Jian Yang**[1]
[1]University of Science and Technology of China
[2] The University of Melbourne [3] Sun Yat-sen University
[4]Institute of Artificial Intelligence, Hefei Comprehensive National Science Center
{ssh0321y, yukariguo}@mail.ustc.edu.cn
yuxiang.wang8@student.unimelb.edu.au

Abstract

Large Reasoning Models (LRMs) achieve strong performance on table reasoning tasks but incur substantial inference cost due to long reasoning traces. Stepwise model routing mitigates this issue by dynamically assigning reasoning steps to smaller or larger models. However, stepwise model routing for table reasoning remains underexplored. Through empirical analysis, we find that reasoning steps involving tables contain two types of tokens with distinct uncertainty distributions: table tokens grounded in table structure, such as cell values and headers, and text tokens representing surrounding natural-language reasoning. The uncertainty of both token types is correlated with the risk that the model makes an error in the next reasoning step. However, existing methods fail to model them separately, leading to suboptimal routing decisions. To address this, we propose EcoTab, a table-aware stepwise routing framework for efficient table reasoning. At each reasoning step, EcoTab separately estimates the uncertainties of table tokens and text tokens, maps them to next-step failure risks for the small model, and combines the two risks for routing. Experiments on multiple table reasoning benchmarks show that EcoTab consistently outperforms strong baselines and achieves a better balance between accuracy and efficiency.

1 Introduction

Table reasoning plays a critical role in real-world applications, including data analytics (Zhao et al., 2024), fact verification (Parikh et al., 2020) and scientific reporting (Newman et al., 2024). However, it remains challenging because tables contain complex structures and implicit relations across rows and columns. Recent Large Reasoning Models (LRMs), such as DeepSeek-R1 (Guo et al., 2025) and OpenAI's o-series (Pfister and Jud, 2025), improve performance by using test-time scaling to produce long reasoning chains during inference. Despite their strong results, this process introduces high computational overhead. The large model size and heavy token usage make LRMs difficult to deploy for table reasoning in latency-sensitive and resource-constrained settings (Zeng et al., 2026).

To mitigate this bottleneck, stepwise model routing (Shi et al., 2025; Lee et al., 2026) has emerged as a promising direction. It decomposes the inference process into multiple reasoning steps, allocating simpler steps to smaller, cheaper models and more complex ones to larger, more expensive models. In this way, stepwise model routing offers an effective balance between efficiency and performance (Fernandez et al., 2026). Existing methods, such as SpecCoT (Shi et al., 2025) and SpecReason (Pan et al., 2025), perform well on free-form text reasoning tasks like mathematical reasoning. However, their effectiveness on structured table reasoning tasks remains underexplored.

To examine this question, we revisit several stepwise model routing methods and evaluate their effectiveness on table reasoning tasks, as detailed in Sec. 3. Our analysis reveals a clear gap in the efficiency and performance trade-off between free-form text reasoning and table reasoning. We find that existing methods often misroute "table-specific steps", such as retrieving the relevant subtable for a question or performing numerical operations over tabular content. To understand this failure, we further analyze the root cause and identify a key insight: such steps contain two types of tokens with distinct uncertainty distributions, namely table tokens, grounded in table structure such as cell values and headers, and text tokens, which reflect the surrounding natural language reasoning. We further find that the uncertainty of both token types is correlated with the next-step failure risk of the small model, which in turn is informative for model selection. However, existing methods lack joint modeling of table tokens and text tokens, resulting in

[*]Equal contribution
[†]Corresponding authors

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 28102–28120
October 24-29, 2026 ©2026 Association for Computational Linguistics

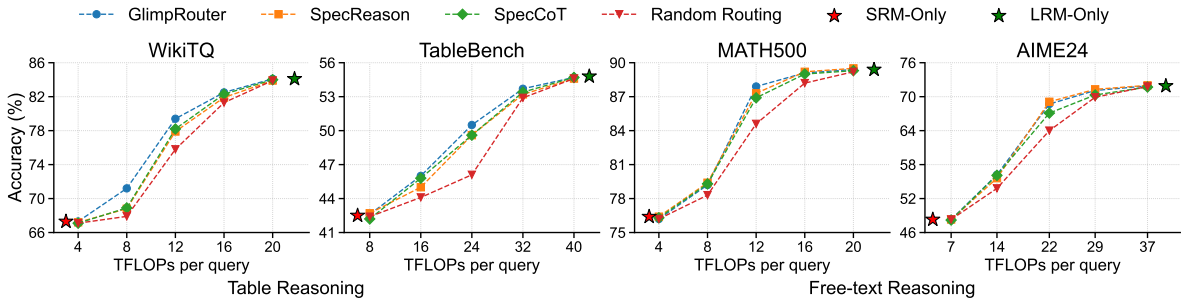

Figure 1: Effectiveness analysis on table and free-form text reasoning tasks. WikiTQ and TableBench represent table reasoning benchmarks, while MATH500 and AIME24 correspond to free-form text reasoning benchmarks.

poor routing performance on table reasoning tasks.

Motivated by these findings, we propose **EcoTab**, an **e**ffi**c**ient table-aware stepwise model r**o**uting framework for **tab**le reasoning. EcoTab is built on a simple intuition: table tokens and text tokens exhibit different uncertainty distributions and should therefore be modeled separately during routing. For each reasoning step, EcoTab first identifies table tokens and text tokens in the current reasoning step and estimates their uncertainties separately. To account for their distinct distributions, EcoTab constructs two offline risk mappings that convert these uncertainties into next-step failure risks, where each risk reflects the likelihood that the small model will fail on the next step. Finally, EcoTab combines the two failure risks into a unified routing score and compares it with a threshold to decide whether the next step should be generated by a small model or a large one.

Our Contributions. (1) *New Perspective*. We conduct the first systematic study of stepwise model routing for table reasoning. We reveal that table reasoning steps contain two types of tokens with distinct uncertainty distributions, explaining why existing routing methods designed for free-form text fail on table reasoning. (2) *Novel Framework*. We propose EcoTab, an efficient table-aware stepwise model routing framework for table reasoning. (3) *SOTA Performance*. Experiments on multiple table reasoning benchmarks show that EcoTab consistently outperforms strong baselines and achieves a better balance between accuracy and efficiency.

2 Preliminary

Table Reasoning with LRMs. Given a table T and a natural language query Q, an LRM generates a sequence of reasoning steps $s_1, \ldots, s_n$, denoted as $s_{1:n}$, where each step s_i contains k_i tokens. Following prior studies (Pan et al., 2025; Zeng et al., 2026), we segment reasoning traces into steps using the newline delimiter "\n\n".

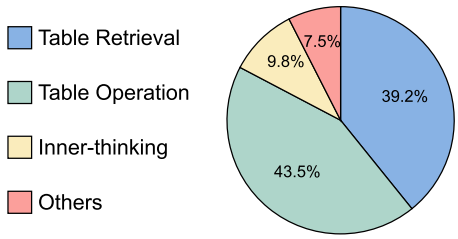

Figure 2: Error distribution across four step categories over 1000 incorrect cases under GlimpRouter.

Stepwise Model Routing. Given the current reasoning prefix $s_{1:n}$, stepwise model routing dynamically chooses between the small reasoning model (SRM) and the large reasoning model (LRM) to generate the next reasoning step s_{i+1}, so as to improve computational efficiency. Formally, the next step is generated as

$$s_{i+1} \sim p_{\theta_{i+1}}(\cdot \mid T, Q, s_{1:i}), \theta_{i+1} = r(\mathcal{I}_{i+1}) \quad (1)$$

where $p_{\theta_{i+1}}$ denotes the probability distribution of the selected reasoning model, and $\theta_{i+1} \in \{\theta_M, \theta_m\}$ corresponds to the LRM or the SRM. Here, $r(\cdot)$ denotes the routing function, and $\mathcal{I}_{i+1}$ denotes the routing information used to determine the model for step $i+1$. Depending on the routing function, $\mathcal{I}_{i+1}$ may come from the previously generated step s_i (Lee et al., 2026), or from a lightweight preview or draft of the next step (Pan et al., 2025; Zeng et al., 2026; Shi et al., 2025). In this work, we use s_i as $\mathcal{I}_{i+1}$, which avoids additional token generation overhead.

3 Motivation

In this section, we investigate why table reasoning requires a dedicated stepwise model routing beyond existing methods designed for free-form text reasoning. This leads to our first research question:

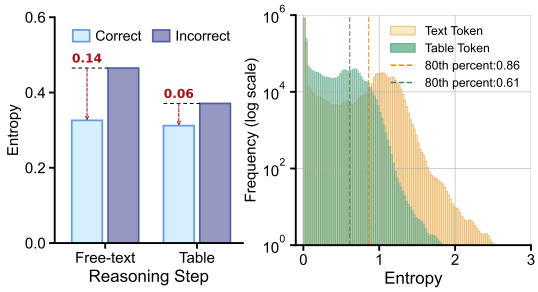

Figure 3: (Left) Difference in average entropy between correct and incorrect steps for Free-Text steps and Table-specific steps. (Right) Entropy distributions of table tokens and text tokens within Table-specific steps.

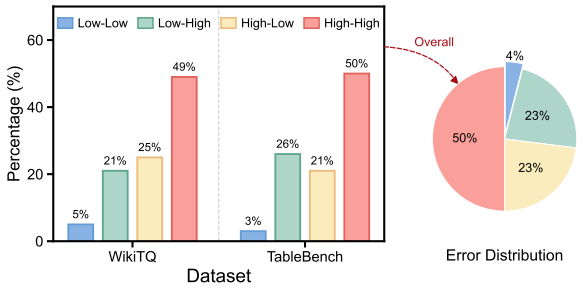

Figure 4: (Left) Error percentages across four entropy groups on WikiTQ and TableBench for Table-specific steps. (Right) Overall error distribution across the four groups.

RQ1 – Do free-form text routing methods adapt effectively to table reasoning?

Effectiveness Analysis. The effectiveness of stepwise model routing depends on reaching LRM-only accuracy at lower cost, which we measure by FLOPs. We evaluate several representative methods, including GlimpRouter (Zeng et al., 2026), SpecReason (Pan et al., 2025), SpecCoT (Shi et al., 2025), and Random Routing. We adopt Qwen3-1.7B and Qwen3-14B (Yang et al., 2025a) as the SRM and LRM. The evaluation covers table reasoning benchmarks, including WikiTQ (Pasupat and Liang, 2015) and TableBench (Wu et al., 2025a), as well as free-form text reasoning benchmarks such as MATH500 (Lightman et al., 2023) and AIME24. More experimental details are provided in Appendix A. As shown in Figure 1, assigning more steps to the stronger LRM naturally increases both FLOPs and accuracy. However, existing routing methods are much less efficient on table reasoning than on free-form text reasoning. On free-form text benchmarks, they achieve near-LRM performance with only about 60% of the full FLOPs. In contrast, on table reasoning benchmarks, they require nearly 80% of the full FLOPs to reach a similar level of performance. This efficiency gap indicates that existing methods do not transfer effectively to table reasoning.

Error Analysis. To understand the source of this efficiency gap, an error analysis of the routing process is conducted. Specifically, we randomly sample 1000 erroneous cases that are correctly solved under the LRM-only setting but fail under GlimpRouter (Zeng et al., 2026), and ask human experts to identify the failure step in each case and classify it into one of four error types, following TaTToo (Zou et al., 2026): (i) *Table Retrieval*, (ii) *Table Operation*, (iii) *Inner-Thinking*, and (iv) *Others* (defined in Appendix A). *Table Retrieval* and *Table Operation* are defined as "table-specific steps", as they are unique to table reasoning, while *Inner-Thinking* and *Others* are regarded as "free-text steps". As shown in Figure 2, 82.7% of routing errors arise from table-specific steps. This indicates that free-form text routing methods fail to properly route these steps, often assigning them to the SRM when the LRM is actually needed.

Finding of RQ1. Free-form text routing methods fail to properly route table-specific steps, leading to a notable efficiency gap in table reasoning.

It raises a subsequent research question:

RQ2 – Why do free-form text routing methods fail on table-specific reasoning steps?

Table Tokens Differ from Text Tokens. To understand this failure, we compare free-text steps with table-specific steps. Following GlimpRouter, we use the average step entropy as the routing score and randomly sample 500 correct steps and 500 incorrect steps. As shown in Figure 3 (left), free-text steps show a clear separation between correct and incorrect cases, with an average entropy gap of 0.14. In contrast, the gap for table-specific steps drops sharply to 0.06. This suggests that step-level entropy is much less informative for routing table-specific steps. We then analyze table-specific steps at the token level by separating each step into *table tokens* and *text tokens*. As shown in Figure 3 (right), the two token types exhibit clearly different entropy distributions. This suggests that they play different roles during reasoning, which is also consistent with prior studies on table reasoning (Wang et al., 2025a; Zou et al., 2026; Li et al., 2025a).

Both Token Types Matter for Routing. Building on the analysis above, we further ask whether both table tokens and text tokens are related to the next-step failure risk. Specifically, we compute the

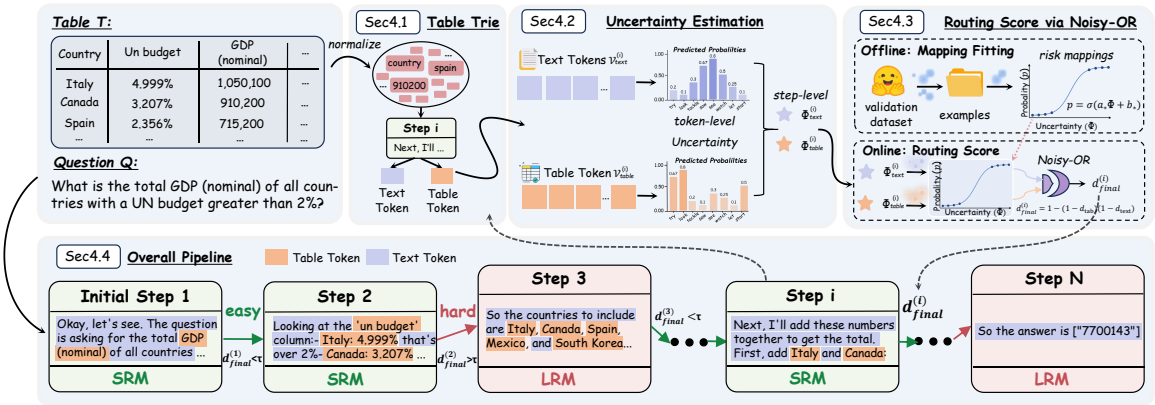

Figure 5: Overview of the EcoTab framework. By separately modeling table tokens and text tokens in each reasoning step s_i, EcoTab enables more effective routing between the SRM and LRM for table reasoning.

average entropy of table tokens and text tokens for 1,000 sampled steps, including 500 correct steps and 500 incorrect steps. For each type of token, we use the 70th percentile as the threshold (Notin et al., 2021). A score above the threshold is labeled as High, and a score below it is labeled as Low. This gives four groups, namely High-High, High-Low, Low-High, and Low-Low. We then examine how the 500 incorrect steps are distributed across these four groups. As shown in Figure 4, errors are not concentrated only in the High-High group. A substantial portion also falls into the High-Low and Low-High groups. This shows that either high table-token uncertainty or high text-token uncertainty alone can be associated with failure risk. Therefore, effective routing should consider both token types rather than relying on only one of them.

Insight for EcoTab. In table reasoning, table tokens and text tokens exhibit different uncertainty distributions, and both are informative of the SRM's next-step failure risk.

4 EcoTab

Motivated by this insight, we propose **EcoTab**, an efficient table-aware stepwise model routing framework for table reasoning. EcoTab works in three stages. Given the input table T, it first builds a lightweight word-level Table Trie to identify table tokens and separate them from text tokens in each reasoning step s_i (Sec. 4.1). For each s_i, EcoTab then estimates table-token uncertainty $\Phi_{\text{tab}}^{(i)}$ and text-token uncertainty $\Phi_{\text{text}}^{(i)}$ (Sec. 4.2). Finally, EcoTab constructs two offline risk mappings, maps the two uncertainties into two failure risks, and combines them into a final routing score $d_{\text{final}}^{(i)}$, which determines whether the next step s_{i+1} should be generated by the SRM or the LRM (Sec. 4.3).

4.1 Table Trie Construction

To separate table tokens from text tokens, we build a word-level Table Trie from the input table T. The Trie stores normalized table content, including column headers and cell values. Before insertion, we apply simple normalization such as lowercasing, removing extra spaces, and standardizing numbers and punctuation. For each reasoning step $s_i = (t_{i,1}, t_{i,2}, \ldots, t_{i,k_i})$, we apply the same normalization and scan the step text from left to right with longest-prefix matching over the Trie. If a span matches a column header or cell value, we mark it as table-related tokens. We then map the matched spans back to token positions and obtain a boolean mask $\mathbf{m}^{(i)} \in \{0,1\}^{k_i}$, where $\mathbf{m}_j^{(i)} = 1$ indicates that $t_{i,j}$ is a table token. Based on this mask, we divide the tokens of s_i into a table-token set $\mathcal{V}_{\text{tab}}^{(i)}$ and a text-token set $\mathcal{V}_{\text{text}}^{(i)}$:

$$\mathcal{V}_{\text{tab}}^{(i)} = \{t_{i,j} \mid \mathbf{m}_j^{(i)} = 1\}, \mathcal{V}_{\text{text}}^{(i)} = \{t_{i,j} \mid \mathbf{m}_j^{(i)} = 0\}, \tag{2}$$

where $1 \leq j \leq k_i$. This procedure is model-free and introduces little overhead in practice.

4.2 Step-Level Uncertainty Estimation

Prior studies have shown that reasoning correctness is closely related to model uncertainty (Xie et al., 2023; Wang and Zhou, 2024). In particular, the uncertainty of the current step s_i can serve as a useful signal for judging whether the current model has sufficient capability to generate the next step correctly (Lee et al., 2026). Specifically, we use Shannon entropy (Shannon, 1948) as the uncertainty measure for each reasoning step. Each step is generated by the current model, which can be either the LRM or the SRM. To quantify the uncertainty of a reasoning step, we first define token-level uncertainty for each token in the step.

Token-Level Uncertainty. For the j-th token $t_{i,j}$ in reasoning step s_i, we define its token-level uncertainty $c_{i,j} \in \mathbb{R}$ as

$$c_{i,j} = -\sum_{v \in \mathcal{V}} p_{i,j}(v) \log p_{i,j}(v), \tag{3}$$

where $\mathcal{V}$ denotes the model vocabulary, $v \in \mathcal{V}$ is a candidate token set, and $p_{i,j}(v)$ denotes the predicted probability of token set v at position (i, j).

Step-Level Uncertainty. For each reasoning step s_i, we use the Table Trie to partition its tokens into a table-token set $\mathcal{V}_{\text{tab}}^{(i)}$ and a text-token set $\mathcal{V}_{\text{text}}^{(i)}$. We then average the token-level uncertainty separately over the two sets to obtain the table uncertainty $\Phi_{\text{tab}}^{(i)}$ and text uncertainty $\Phi_{\text{text}}^{(i)}$:

$$\Phi_*^{(i)} = \frac{1}{|\mathcal{V}_*^{(i)}|} \sum_{t_{i,j} \in \mathcal{V}_*^{(i)}} c_{i,j}, \quad * \in \{\text{tab}, \text{text}\}. \tag{4}$$

4.3 Routing Score via Noisy-OR

Given table-token uncertainty $\Phi_{\text{tab}}^{(i)}$ and text-token uncertainty $\Phi_{\text{text}}^{(i)}$, we need to combine them into a single routing score for model selection. However, the two uncertainties follow different distributions. If they are directly averaged or linearly weighted, the resulting score can be poorly calibrated and less reliable for routing. To make them more comparable, we first map them into next-step failure risks in $[0, 1]$. For each uncertainty $\Phi_*^{(i)}$, where $* \in \{\text{tab}, \text{text}\}$, we define $d_*^{(i)} \approx \Pr(\text{SRM fails on } s_{i+1} \mid \Phi_*^{(i)})$, which measures how likely the SRM is to fail if it is used to generate the next step. To estimate this mapping, we build two offline failure-risk mappings on a held-out validation set. Specifically, we construct step-level supervision by identifying a critical routing boundary for each retained sample, namely the step whose routing score should trigger switching to the LRM for generating the next step. This boundary is determined through suffix replacement on validation trajectories, as detailed in Appendix C. The identified boundary step is labeled as positive, earlier retained steps are labeled as negative, and later steps are discarded. This construction aligns the supervision target with the next-step routing objective. We then fit a sigmoid mapping for each uncertainty:

$$d_*^{(i)} = f_*\left(\Phi_*^{(i)}\right) = \sigma\left(a_* \Phi_*^{(i)} + b_*\right), \tag{5}$$

where a_* and b_* are learned from the validation set. After calibration, the two failure risks are combined using Noisy-OR (Pearl, 2014):

$$d_{\text{final}}^{(i)} = 1 - \left(1 - d_{\text{tab}}^{(i)}\right)\left(1 - d_{\text{text}}^{(i)}\right). \tag{6}$$

Finally, we compare $d_{\text{final}}^{(i)}$ with a threshold τ to decide whether the next step s_{i+1} should be generated by the LRM or the SRM.

4.4 Overall Pipeline

The overall procedure distinguishes the initial step from all subsequent steps, since no prior routing score is available for the first step. Here, θ_M and θ_m denote the LRM and the SRM, respectively.

(1) *Initial step*. For all samples, θ_m first generates the first reasoning step s_1.

(2) *First-step refinement*. Compute the routing score $d_{\text{final}}^{(1)}$ of s_1, and regenerate the first step s_1 by θ_M if $d_{\text{final}}^{(1)} > \tau$.

(3) *Iterative*. For each s_i, the routing score $d_{\text{final}}^{(i)}$ is used to choose θ_M or θ_m for the next step s_{i+1}.

(4) *Answer generation*. Repeat step (3) until the final answer is produced or reach the iteration limit.

5 Experiments

In this section, we present a comprehensive evaluation of EcoTab. We first describe the experimental setup and evaluation metrics (Sec. 5.1). We then organize our experiments around four key research questions: **Q1**: Can EcoTab consistently outperform existing state-of-the-art stepwise model routing methods on table reasoning tasks (Sec. 5.2)? **Q2**: How does each core component of EcoTab contribute to its overall performance (Sec. 5.3)? **Q3**: Is EcoTab robust to different calibration and threshold settings (Sec. 5.4)? **Q4**: Can EcoTab effectively capture reasoning difficulty while remaining lightweight in routing overhead (Sec. 5.5)?

5.1 Experimental Setup

Models and Configurations. We use Qwen3-1.7B (Yang et al., 2025a) as the SRM, and evaluate two LRMs: Qwen3-14B (Yang et al., 2025a) and DeepSeek-R1-Distill-Qwen-14B (Guo et al., 2025). This setup allows us to study both same-family and cross-family collaboration.

Benchmarks. We evaluate EcoTab on five table reasoning benchmarks. TableBench (Wu et al., 2025a) contains 886 complex questions spanning numerical reasoning, fact checking, and data analysis. WikiTQ (Pasupat and Liang, 2015) focuses on question answering over Wikipedia tables, and TabFact (Chen et al., 2020a) evaluates table-based

Method	Extra Tokens	WikiTQ		TabFact		TableBench		HiTab		FinQA		Average		
		Acc ↑	FLOPs ↓	Acc ↑	FLOPs ↓	Acc ↑	FLOPs ↓	Acc ↑	FLOPs ↓	Acc ↑	FLOPs ↓	Acc ↑	FLOPs ↓	A/F ↑
Qwen3-Instruct														
1.7B Only	–	67.34	3.84	82.42	2.51	42.55	6.12	59.34	4.82	47.52	7.38	59.83	4.93	15.2
14B Only	–	84.12	21.6	90.47	14.6	54.85	42.4	80.17	20.5	67.74	48.0	75.47	29.4	3.34
Random	✗	75.82	18.5	86.76	10.6	46.12	36.7	73.43	17.4	58.78	40.4	68.18	24.7	3.84
RSD	✓	79.73	16.2	88.24	9.32	49.21	30.4	75.88	15.2	63.87	36.4	71.39	21.5	4.55
SpecCoT	✓	78.50	15.8	87.98	8.98	49.87	29.7	76.04	15.0	64.01	36.0	71.28	21.1	4.66
SpecReason	✓	78.76	15.7	87.74	9.21	50.02	29.9	76.41	15.0	64.21	36.0	71.43	21.1	4.62
STEER	✗	79.46	15.4	88.44	8.43	50.33	29.4	77.04	14.9	63.73	35.4	71.80	20.7	4.87
GlimpRouter	✓	79.04	16.4	88.85	8.56	50.52	30.1	76.22	15.4	63.98	36.1	71.72	21.3	4.72
EcoTab(ours)	✗	**80.83**	**14.0**	**89.14**	**8.42**	**51.77**	**28.1**	**78.32**	**13.9**	**65.48**	**34.3**	**73.11**	**19.7**	**5.15**
DeepSeek-R1-Distill-Qwen														
1.7B Only	–	67.34	3.84	82.42	2.51	42.55	6.12	59.34	4.82	47.52	7.38	59.83	4.93	15.22
14B Only	–	81.71	20.8	89.21	15.3	53.65	40.5	70.08	22.9	69.40	30.9	72.81	26.0	3.28
Random	✗	74.78	17.9	85.77	11.1	44.03	36.4	63.06	18.1	59.67	26.7	65.46	22.0	3.77
RSD	✓	78.25	15.3	87.12	9.01	48.81	28.6	66.67	15.3	66.02	21.9	69.37	18.0	4.77
SpecCoT	✓	78.12	15.0	87.41	8.86	48.74	28.3	66.94	15.0	65.89	22.6	69.42	17.9	4.83
SpecReason	✓	78.01	15.0	87.57	8.94	48.89	28.4	66.58	15.1	66.07	22.2	69.42	17.9	4.82
STEER	✗	78.32	14.4	87.64	8.83	49.22	29.1	67.01	15.3	66.15	**21.3**	69.67	17.8	4.91
GlimpRouter	✓	78.04	14.8	87.81	**8.78**	49.54	28.1	66.71	14.8	65.81	22.8	69.58	17.8	4.89
EcoTab(ours)	✗	**79.56**	**13.1**	**88.04**	8.81	**50.41**	**27.4**	**67.97**	**13.9**	**67.63**	21.4	**70.72**	**16.9**	**5.19**

Table 1: Main Results. Acc ↑ is measured at 60% of LRM-only FLOPs, and FLOPs ↓ denotes the computation required to reach 98% of LRM-only accuracy. The best result is **bold**, and the second-best result is underlined. Extra Tokens denotes whether the method requires additional token generation during the reasoning process.

fact verification. To further test robustness, we include HiTab (Cheng et al., 2022), which features hierarchical nested tables, and FinQA (Chen et al., 2021), a text+table reasoning dataset requiring joint understanding of financial reports and tabular data. Additional details are provided in Appendix A.2.

Baselines. We compare EcoTab against standalone models and state-of-the-art stepwise model routing baselines, including SRM/LRM Only, Random, RSD (Liao et al., 2025), SpecCoT (Shi et al., 2025), SpecReason (Pan et al., 2025), STEER (Lee et al., 2026) and GlimpRouter (Zeng et al., 2026).

Evaluation Metric. Following prior work (Liao et al., 2025; Sardana et al., 2024), we use accuracy as the performance metric and estimate inference cost using the standard Transformer FLOPs approximation of $2N$ per generated token for a model with N parameters. To enable clearer comparison across baselines, we conduct a grid search over threshold values with a step size of 0.05 for all baseline methods and EcoTab, and derive the corresponding accuracy–FLOPs trade-off curves. We then report the accuracy achieved at 60% of the LRM-only FLOPs, as well as the FLOPs required to reach 98% of the LRM-only accuracy. In addition, we report Accuracy-per-FLOPs (A/F), adapted from (Ma et al., 2025), to better characterize the trade-off between performance and computational cost.

Method	Acc ↑	▽	FLOPs ↓	▽
EcoTab (Qwen3-Instruct)	**51.8**	–	**28.1**	–
w/ average token	49.9	(-1.9)	30.3	(+2.2)
w/ only table token	50.5	(-1.3)	29.4	(+1.3)
w/ only text token	50.2	(-1.6)	29.7	(+1.6)
w/ linear weighting	50.0	(-1.8)	29.8	(+1.7)
EcoTab (DeepSeek-R1)	**50.4**	–	**27.4**	–
w/ average token	48.9	(-1.5)	30.0	(+2.6)
w/ only table token	49.6	(-0.8)	28.3	(+0.9)
w/ only text token	49.4	(-1.0)	28.6	(+1.2)
w/ linear weighting	49.1	(-1.3)	29.0	(+1.6)

Table 2: Ablation results of EcoTab on TableBench. We report accuracy (Acc) and the average FLOPs per query.

5.2 Main Results

Table 1 presents the main results of EcoTab on five table reasoning benchmarks under both same-family and cross-family collaboration settings. Overall, EcoTab consistently outperforms all baselines and achieves the best accuracy–efficiency trade-off in both settings. In particular, it obtains the best overall average performance in terms of accuracy, FLOPs, and A/F, showing that EcoTab can improve reasoning quality while reducing inference cost. A closer look shows that EcoTab remains consistently effective across different datasets and model combinations. Under both Qwen3-14B and DeepSeek-R1-Distill-Qwen-14B as the LRM, EcoTab achieves the highest accuracy on all five benchmarks, while also maintaining the lowest overall inference cost. This result verifies that the advantage of EcoTab is not limited to a specific model family, but generalizes to both same-family and cross-family collaboration. Another notable

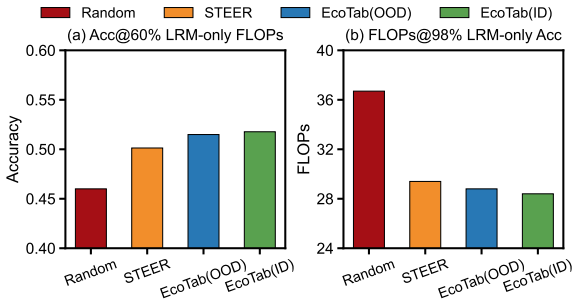

Figure 6: Failure-risk mapping transferability of EcoTab. ID denotes in-domain evaluation, and OOD denotes out-of-domain transfer.

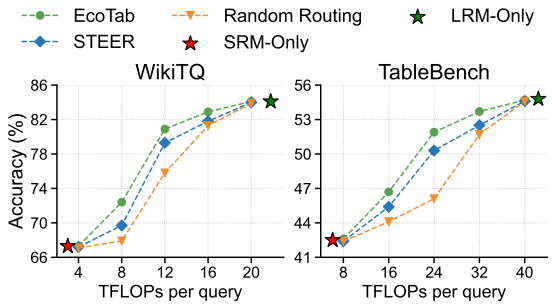

Figure 7: Threshold robustness of EcoTab on WikiTQ and TableBench. As the threshold τ increases, the total FLOPs decrease.

advantage is that EcoTab does not require extra token generation during routing, yet it still surpasses strong baselines that rely on verification overhead. This suggests that explicitly separating table-token and text-token is more effective for table reasoning than directly applying free-form text routing methods. Overall, the results confirm that EcoTab is a more suitable step-level routing framework for table reasoning, delivering a stronger balance between effectiveness and efficiency.

5.3 Ablation Study

Table 2 reports the ablation results of EcoTab on TableBench. Overall, the full EcoTab consistently performs best under both Qwen3-Instruct and DeepSeek-R1 settings, achieving the highest accuracy with the lowest FLOPs. Removing the separation between table tokens and text tokens (*w/ average token*) causes the largest performance drop and a clear increase in inference cost, showing that collapsing all tokens into a single score weakens routing decisions. Using only table tokens (*w/ only table token*) or only text tokens (*w/ only text token*) also degrades performance, indicating that both types of token are necessary and complementary for effective routing. In addition, replacing Noisy-OR with simple linear weighting (*w/ linear weighting*) consistently hurts both accuracy and efficiency, suggesting that EcoTab benefits not only from separating the two token types, but also from using a more suitable fusion strategy. Overall, these results indicate that all components of EcoTab are essential to its effectiveness.

5.4 Robustness Analysis

Failure-risk mapping transferability. We study whether the fitted score-to-risk mapping in Sec. 4.3 can transfer across domains. Specifically, we compare an in-domain setting (ID), where the mapping is fitted on the target dataset, with an out-of-domain setting (OOD), where the mapping fitted on the WikiTQ validation split is directly applied to TableBench. Figure 6 shows that the OOD variant still outperforms Random and STEER on both metrics. It achieves higher accuracy at 60% of LRM only FLOPs and requires fewer FLOPs to reach 98% of LRM only accuracy. Although the ID setting performs slightly better, the gap is small, which suggests that the learned risk mapping generalizes well across table reasoning domains.

Threshold robustness. We further vary the routing threshold and plot the accuracy–FLOPs curves on WikiTQ and TableBench. A higher threshold makes the router more conservative in calling the LRM, so fewer steps are assigned to the LRM and the total FLOPs become lower. As shown in Figure 7, EcoTab consistently stays above STEER and Random on both datasets. It achieves higher accuracy at similar cost, or lower cost at similar accuracy. Overall, the gain comes from more reliable model routing rather than a specific threshold.

5.5 Further Analysis

Difficulty awareness. We further examine whether EcoTab is sensitive to sample difficulty. Following prior work (Ye et al., 2025), we estimate TableBench difficulty using GPT-5.4-High. For each question, we sample 100 independent answers and use the number of correct responses as a proxy for difficulty. Based on this score, we group samples into three subsets: *hard* (0–9 correct), *medium* (10–59 correct), and *easy* (60–100 correct). We then compute the LRM usage rate of EcoTab at different relative step positions within each group. Figure 8 shows that harder samples consistently trigger more LRM calls than easier ones. This pattern holds for both correct and incorrect samples, and is especially clear on incorrect

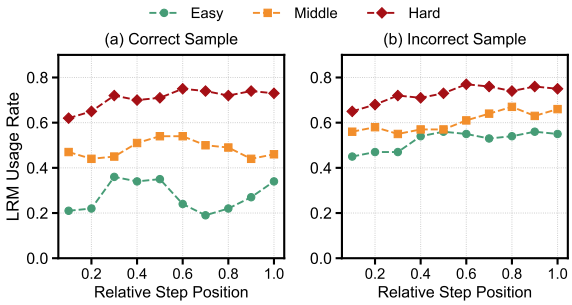

Figure 8: LRM usage rate across GPT-5.4-High difficulty levels, comparing correct and incorrect samples.

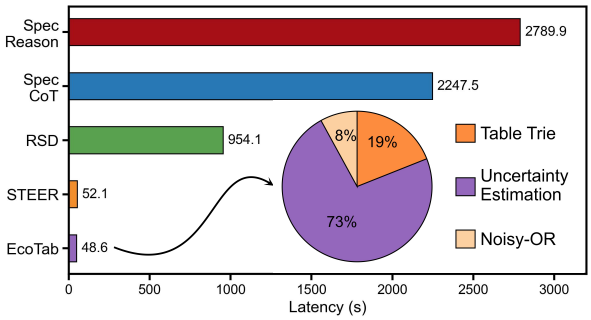

Figure 9: Overall routing latency on TableBench, together with the latency breakdown of EcoTab.

samples. These results suggest that EcoTab can capture sample-level difficulty and allocate more LRM computation to harder cases.

Routing overhead. We then compare the routing latency of different methods on the full TableBench benchmark, excluding the reasoning time and measuring only model routing overhead. Figure 9 shows that EcoTab has an overall latency of 48.6 seconds, which is comparable to STEER at 52.1 seconds, while remaining far below RSD, SpecCoT, and SpecReason. This result shows that EcoTab introduces little extra overhead and remains a lightweight routing method in practice.

6 Related Work

Table Reasoning. Reasoning over tables poses unique challenges for LLMs, as it requires both natural language understanding and structured reasoning over rows, columns, and cell values (Jin et al., 2022; Zhang et al., 2025). Recent studies (Chen et al., 2020b; Deng et al., 2022; Iida et al., 2021) have explored table reasoning across a variety of downstream tasks, including table qa (Chen et al., 2020b) and table fact verification (Parikh et al., 2020). Early methods, such as TAPAS (Herzig et al., 2020) and TaBERT (Yin et al., 2020), mainly model tables through Transformer-based encoders. With the rise of LLMs, later approaches began to improve table reasoning through prompt engineering (Sui et al., 2024; Wang et al., 2024) or supervised fine-tuning (Su et al., 2024). More recent works, such as the Table-R1 series (Yang et al., 2025b; Wu et al., 2025b; Jin et al., 2025), further enhance reasoning performance by using reinforcement learning to optimize reasoning trajectories. However, as table size grows and reasoning models become larger and more verbose, inference latency becomes increasingly prohibitive. EcoTab addresses this issue by introducing a table-aware step-level routing framework to better balance efficiency and performance.

Efficient Reasoning. Scaling test-time compute in LRMs can substantially improve reasoning performance, but also incurs prohibitive latency (Wang et al., 2025b). To mitigate this cost, recent work has explored dynamically offloading part of the reasoning process to smaller models, mainly at three levels: query-level routing (Chen et al., 2023b; Wang et al., 2025c; Zhao et al., 2025a), step-level routing (Shi et al., 2025; Pan et al., 2025; Lee et al., 2026), and token-level speculation (Leviathan et al., 2023; Chen et al., 2023a). Among them, step-level routing is particularly well aligned with the multi-step reasoning nature of LRMs. Existing methods range from training-based reward guidance, such as RSD (Liao et al., 2025), to training-free paradigms based on multi-path selection, such as SpecCoT (Shi et al., 2025), post-hoc verification, such as SpecReason (Pan et al., 2025), and methods that route models based on uncertainty signals, such as entropy (Zeng et al., 2026) or confidence (Lee et al., 2026). However, these methods are mainly designed for free-form text reasoning and do not account for the structured nature of table reasoning.

7 Conclusion

In this paper, we study efficient stepwise model routing for table reasoning with LRMs. We show that existing step-level routing methods designed for free-form text reasoning are less effective in tabular settings due to the structured nature of tables. To address this issue, we propose EcoTab, an adaptive table-aware step-level routing framework that explicitly distinguishes table tokens and text tokens when estimating step difficulty. The resulting routing score is computed through a probabilistic fusion mechanism to guide model selection during reasoning. Experiments on multiple table reasoning benchmarks demonstrate that EcoTab consis-

tently achieves a better trade-off between reasoning accuracy and computational cost compared with existing stepwise model routing methods. These results highlight the importance of table-aware routing for efficient reasoning over structured data.

Limitations

Like existing routing methods, EcoTab cannot explicitly control the output length of the reasoning process. In practice, the routing decision only determines which model generates the next step, but does not decide when the reasoning should stop. As a result, unnecessary long reasoning traces may still appear even when the routing is accurate. A promising direction for future work is to design a table aware early stopping mechanism that can terminate reasoning once the required table evidence and logical deductions are sufficient.

Ethics Statement

Our work aims to improve the efficiency and reliability of multi-step table reasoning through stepwise model routing. However, like any system built on LLMs, it may still produce incorrect intermediate reasoning steps or factually incorrect final answers. We therefore encourage users to exercise caution and verify critical outputs when deploying such systems in real-world scenarios. Furthermore, our research builds upon open-source models and frameworks, including Qwen3, DeepSeek-R1-Distill, PyTorch, and Hugging Face. We strictly follow their respective licenses and usage policies, and acknowledge their important contributions to the research community.

Acknowledgement

This work is supported by the Fundamental Research Funds for the Central Universities under Grant No. WK2102026004 and Anhui Provincial Science and Technology Innovation Key Research Program (202523j08050016).

References

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

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023a. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Lekai Chen, Ashutosh Trivedi, and Alvaro Velasquez. 2024. Llms as probabilistic minimally adequate teachers for dfa learning. *arXiv preprint arXiv:2408.02999*.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023b. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020a. Tabfact: A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations*.

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

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

Zhuohong Chen, Zhenxian Wu, Yunyao Yu, Hangrui Xu, Zirui Liao, Zhifang Liu, Xiangwen Deng, Pen Jiao, and Haoqian Wang. 2026. Learning to search: A decision-based agent for knowledge-based visual question answering. *Preprint*, arXiv:2604.07146.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2022. HiTab: A hierarchical table dataset for question answering and natural language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1094–1110, Dublin, Ireland. Association for Computational Linguistics.

Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2022. Turl: Table understanding through representation learning. *ACM SIGMOD Record*, 51(1):33–40.

Nigel Fernandez, Branislav Kveton, Ryan A. Rossi, Andrew Lan, and Zichao Wang. 2026. RADAR: Reasoning-ability and difficulty-aware routing for reasoning LLMs. In *The Fourteenth International Conference on Learning Representations*.

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

Zhezheng Hao, Hong Wang, Haoyang Liu, Jian Luo, Jiarui Yu, Hande Dong, Qiang Lin, Can Wang, and Jiawei Chen. 2026a. Rethinking entropy interventions in rlvr: An entropy change perspective. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 31105–31133.

Zhezheng Hao, Tianfu Wang, Huanshuo Dong, Ziyan Liu, Hong Wang, Xiankun Lin, Qiang Lin, Can Wang, Hande Dong, and Jiawei Chen. 2026b. Evolve as a team: Collaborative self-evolution for llm-based multi-agent systems. *arXiv preprint arXiv:2605.29790*.

Zhezheng Hao, Haonan Xin, Long Wei, Liaoyuan Tang, Rong Wang, and Feiping Nie. 2024. Towards expansive and adaptive hard negative mining: Graph contrastive learning via subspace preserving. In *Proceedings of the ACM Web Conference 2024*, pages 322–333.

Yingjie He, Zhaolu Kang, Kehan Jiang, Qianyuan Zhang, Jiachen Qian, Chunlei Meng, Yujie Feng, Yuan Wang, Jiabao Dou, Aming Wu, et al. 2026. How order-sensitive are llms? orderprobe for deterministic structural reconstruction. *arXiv preprint arXiv:2601.08626*.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 4320–4333.

Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. Tabbie: Pretrained representations of tabular data. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3446–3456.

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.

Rihui Jin, Zheyu Xin, Xing Xie, Zuoyi Li, Guilin Qi, Yongrui Chen, Xinbang Dai, Tongtong Wu, and Gholamreza Haffari. 2025. Table-r1: Self-supervised and reinforcement learning for program-based table reasoning in small language models. *arXiv preprint arXiv:2506.06137*.

Sangmook Lee, Dohyung Kim, Hyukhun Koh, Nakyeong Yang, and Kyomin Jung. 2026. Confidence-guided stepwise model routing for cost-efficient reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 31483–31491.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Liyao Li, Chao Ye, Wentao Ye, Yifei Sun, Zhe Jiang, Haobo Wang, Jiaming Tian, Yiming Zhang, Ningtao Wang, Xing Fu, et al. 2025a. Table as a modality for large language models. *arXiv preprint arXiv:2512.00947*.

Songze Li, Yarong Lan, Zhongpu Bo, Zhaoyang Wang, Zhiqiang Liu, Yuan Yuan, Chengtao Gan, Menghao Qian, Enpei Niu, Xiaoke Guo, et al. 2026a. Scaling llm knowledge boundaries via distribution-optimized synthesis. *arXiv preprint arXiv:2606.23271*.

Songze Li, Zhiqiang Liu, Zhaoyan Gong, Xiaoke Guo, Zhongpu Bo, Zhengke Gui, Lei Liang, Huajun Chen, and Wen Zhang. 2026b. Last layer logits to logic: Empowering llms with logic-consistent structured knowledge reasoning. *Preprint*, arXiv:2511.07910.

Songze Li, Zhiqiang Liu, Zhengke Gui, Huajun Chen, and Wen Zhang. 2025b. Enrich-on-graph: Query-graph alignment for complex reasoning with LLM enriching. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7672–7692, Suzhou, China. Association for Computational Linguistics.

Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. 2025. Reward-guided speculative decoding for efficient LLM reasoning. In *Forty-second International Conference on Machine Learning*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Weiting Liu, Jieyi Bi, Wanqi Zhou, Jianfeng Feng, Yining Ma, Ai Han, and Wenlian Lu. 2026a. Toolanchor: Anchoring counterfactual context to boost agentic tool-use capability. *arXiv preprint arXiv:2607.14145*.

Weiting Liu, Han Wu, Yufei Kuang, Xiongwei Han, Tao Zhong, Jianfeng Feng, and Wenlian Lu. 2026b. Automated optimization modeling via a localizable error-driven perspective. *arXiv preprint arXiv:2602.11164*.

Yule Liu, Heyi Zhang, Jinyi Zheng, Zhen Sun, Zifan Peng, Jiaheng Wei, Tianshuo Cong, Yilong Yang, and Xinlei He. 2026c. Auditing data membership in reinforcement learning with verifiable rewards. *Preprint*, arXiv:2511.14045.

Yule Liu, Jingyi Zheng, Zhen Sun, Zifan Peng, Wenhan Dong, Zeyang Sha, Shiwen Cui, Weiqiang Wang, and Xinlei He. 2025. Thought manipulation: External thought can be efficient for large reasoning models. *Preprint*, arXiv:2504.13626.

Ziyan Liu, Zhezheng Hao, Yeqiu Chen, Hong Wang, Jingren Hou, Ruiyi Ding, Yongkang Yang, Wence Ji, Wei Xia, and Feng Liu. 2026d. Meta-cognitive memory policy optimization for long-horizon llm agents. *arXiv preprint arXiv:2605.30159*.

Chunji Lv, Yangguang Wei, Junlin Liu, Yang Gao, Ming Liu, Xinming Wang, Jinyang Wu, Guoren Wang, and Changsheng Li. 2026. Pcsd: Persistent consistency for self-distillation in agentic reinforcement learning. *Preprint*, arXiv:2608.01837.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. Cot-valve: Length-compressible chain-of-thought tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6025–6035.

Benjamin Newman, Yoonjoo Lee, Aakanksha Naik, Pao Siangliulue, Raymond Fok, Juho Kim, Daniel S Weld, Joseph Chee Chang, and Kyle Lo. 2024. Arxivdigestables: Synthesizing scientific literature into tables using language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9612–9631.

Pascal Notin, José Miguel Hernández-Lobato, and Yarin Gal. 2021. Improving black-box optimization in VAE latent space using decoder uncertainty. In *Advances in Neural Information Processing Systems*.

Rui Pan, Yinwei Dai, Zhihao Zhang, Gabriele Oliaro, Zhihao Jia, and Ravi Netravali. 2025. Specreason: Fast and accurate inference-time compute via speculative reasoning. In *NeurIPS 2025 Workshop on Efficient Reasoning*.

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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Judea Pearl. 2014. *Probabilistic reasoning in intelligent systems*, volume 88. Elsevier.

Rolf Pfister and Hansueli Jud. 2025. Understanding and benchmarking artificial intelligence: Openai's o3 is not agi. *arXiv preprint arXiv:2501.07458*.

Nikhil Sardana, Jacob Portes, Sasha Doubov, and Jonathan Frankle. 2024. Beyond chinchilla-optimal: Accounting for inference in language model scaling laws. In *International Conference on Machine Learning*, pages 43445–43460. PMLR.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Junhan Shi, Yijia Zhu, Zhenning Shi, Dan Zhao, Qing Li, and Yong Jiang. 2025. SpecCoT: Accelerating chain-of-thought reasoning through speculative exploration. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 24405–24415, Suzhou, China. Association for Computational Linguistics.

Qiming Shi, Zhaolu Kang, Yunfan Zhou, Di Weng, and Yingcai Wu. 2026a. Spader: Step-wise peer advantage with diversity-aware exploration rewards for multi-answer question answering. *arXiv preprint arXiv:2606.00593*.

Yibo Shi, Jungang Li, Linghao Zhang, Zihao Dongfang, Biao Wu, Sicheng Tao, Yibo Yan, Chenxi Qin, Weiting Liu, Zhixin Lin, et al. 2026b. Androtmem: From interaction trajectories to anchored memory in long-horizon gui agents. *arXiv preprint arXiv:2603.18429*.

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

Yuan Sui, Jiaru Zou, Mengyu Zhou, Xinyi He, Lun Du, Shi Han, and Dongmei Zhang. 2024. Tap4llm: Table provider on sampling, augmenting, and packing semi-structured data for large language model reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10306–10323.

Wenxin Tang, Jingyu Xiao, Yanpei Gong, Fengyuan Ran, Tongchuan Xia, Junliang Liu, Man Ho Lam, Wenxuan Wang, and Michael R Lyu. 2026a. Efficientpostergen: Semantic-aware efficient poster generation via token compression and accurate violation detection. *arXiv preprint arXiv:2603.00155*.

Wenxin Tang, Jingyu Xiao, Zhenyu Liu, Zipeng Xie, Junliang Liu, Wang Luo, Yuan Jiang, Yintong Huo, and Michael Lyu. 2026b. Codeshrink: Adaptive visual compression for efficient multimodal code understanding. *arXiv preprint arXiv:2607.29637*.

Hong Wang, Zhezheng Hao, Jian Luo, Chenxing Wei, Yao Shu, Lei Liu, Qiang Lin, Hande Dong, and Jiawei Chen. 2026. Scheduling your llm reinforcement learning with reasoning trees. In *International Conference on Learning Representations*, volume 2026, pages 154734–154753.

Lanrui Wang, Mingyu Zheng, Hongyin Tang, Zheng Lin, Yanan Cao, Jingang Wang, Xunliang Cai, and

Weiping Wang. 2025a. NeedleinATable: Exploring long-context capability of large language models towards long-structured tables. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Qiuchen Wang, Ruixue Ding, Yu Zeng, Zehui Chen, Lin Chen, Shihang Wang, Pengjun Xie, Fei Huang, and Feng Zhao. 2025b. Vrag-rl: Empower vision-perception-based rag for visually rich information understanding via iterative reasoning with reinforcement learning. *arXiv preprint arXiv:2505.22019*.

Xinyuan Wang, Yanchi Liu, Wei Cheng, Xujiang Zhao, Zhengzhang Chen, Wenchao Yu, Yanjie Fu, and Haifeng Chen. 2025c. Mixllm: Dynamic routing in mixed large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 10912–10922.

Xuezhi Wang and Denny Zhou. 2024. Chain-of-thought reasoning without prompting. *Advances in Neural Information Processing Systems*, 37:66383–66409.

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

Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xeron Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, et al. 2025a. Tablebench: A comprehensive and complex benchmark for table question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 24, pages 25497–25506.

Xingjian Wu, Junlin Liu, Xingchen Liu, Xuhang Zhu, Jianing Wang, Linsen Guo, Xiaoyu Li, Xuezhi Cao, and Xunliang Cai. 2026a. Contrastive reinforced policy optimization via privileged self-distillation. *arXiv preprint arXiv:2607.28026*.

ZhengXian Wu, Hangrui Xu, Kai Shi, Zhuohong Chen, Yunyao Yu, Chuanrui Zhang, Zirui Liao, Jun Yang, Zhenyu Yang, Haonan Lu, et al. 2026b. Promsa: Progressive multimodal search agents for knowledge-based visual question answering. *arXiv preprint arXiv:2606.27974*.

Zhenhe Wu, Jian Yang, Jiaheng Liu, Xianjie Wu, Changzai Pan, Jie Zhang, Yu Zhao, Shuangyong Song, Yongxiang Li, and Zhoujun Li. 2025b. Table-r1: Region-based reinforcement learning for table understanding. *arXiv preprint arXiv:2505.12415*.

Xi Xiao, Chen Liu, Chih-Ting Liao, Yunbei Zhang, Qizhen Lan, Yuxiang Wei, Lin Zhao, Janet Wang, Jianyang Gu, Muchao Ye, et al. 2026a. Staying vigilant: Mitigating visual laziness via counterfactual visual alignment in mllms. *arXiv preprint arXiv:2606.26387*.

Xi Xiao, Chenrui Ma, Yunbei Zhang, Chen Liu, Zhuxuanzi Wang, Yanshu Li, Lin Zhao, Guosheng Hu, Tianyang Wang, and Hao Xu. 2026b. Not all directions matter: Towards structured and task-aware low-rank model adaptation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2132–2154.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. 2023. Self-evaluation guided beam search for reasoning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Chu Xu, Xinke Jiang, Rihong Qiu, Jiaran Gao, and Junfeng Zhao. 2025. Model shapley: Find your ideal parameter player via one gradient backpropagation. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 71150–71200. Curran Associates, Inc.

Zukang Xu, Zhixiong Zhao, Xing Hu, Zhixuan Chen, and Dawei Yang. 2026. Kbvq-moe: Klt-guided svd with bias-corrected vector quantization for moe large language models. *arXiv preprint arXiv:2602.11184*.

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

Zheyuan Yang, Lyuhao Chen, Arman Cohan, and Yilun Zhao. 2025b. Table-r1: Inference-time scaling for table reasoning tasks. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20616–20635.

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*.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 8413–8426.

Yi Yu, Guangquan Hu, Chenghuang Shen, Xingyan Liu, Jing Gu, Hangyi Sun, Junzhuo Ma, Weiting Liu, Jianfeng Liu, Mingyue Pu, et al. 2026a. Cirrusbench: Evaluating llm-based agents beyond correctness in real-world cloud service environments. *arXiv preprint arXiv:2603.28569*.

Yi Yu, Junzhuo Ma, Chenghuang Shen, Xingyan Liu, Jing Gu, Hangyi Sun, Guangquan Hu, Jianfeng Liu, Weiting Liu, Mingyue Pu, et al. 2026b. Lightweight adaptation for llm-based technical service agent: Latent logic augmentation and robust noise reduction. *arXiv preprint arXiv:2603.18074*.

Wenhao Zeng, Xuteng Zhang, Yuling Shi, Chao Hu, Yuting Chen, Beijun Shen, and Xiaodong Gu. 2026.

Glimprouter: Efficient collaborative inference by glimpsing one token of thoughts. *arXiv preprint arXiv:2601.05110*.

Linghao Zhang, Jungang Li, Yonghua Hei, Sicheng Tao, Song Dai, Yibo Yan, Zihao Dongfang, Weiting Liu, Chenxi Qin, Hanqian Li, et al. 2026. Temporal gains, spatial costs: Revisiting video fine-tuning in multimodal large language models. *arXiv preprint arXiv:2603.17541*.

Xuanliang Zhang, Dingzirui Wang, Longxu Dou, Qingfu Zhu, and Wanxiang Che. 2025. A survey of table reasoning with large language models. *Frontiers of Computer Science*, 19(9):199348.

Zhibo Zhang, Wuxia Bai, Yuxi Li, Mark Huasong Meng, Kailong Wang, Ling Shi, Li Li, Jun Wang, and Haoyu Wang. 2024. Glitchprober: Advancing effective detection and mitigation of glitch tokens in large language models. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, ASE '24, pages 643–655, New York, NY, USA. Association for Computing Machinery.

Bo Zhao, Berkcan Kapusuzoglu, Kartik Balasubramaniam, Sambit Sahu, Supriyo Chakraborty, and Genta Indra Winata. 2025a. Optimizing reasoning efficiency through prompt difficulty prediction. *arXiv preprint arXiv:2511.03808*.

Yilun Zhao, Yitao Long, Hongjun Liu, Ryo Kamoi, Linyong Nan, Lyuhao Chen, Yixin Liu, Xiangru Tang, Rui Zhang, and Arman Cohan. 2024. Docmath-eval: Evaluating math reasoning capabilities of llms in understanding long and specialized documents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16103–16120.

Zhixiong Zhao, Haomin Li, Fangxin Liu, Yuncheng Lu, Zongwu Wang, Tao Yang, Li Jiang, and Haibing Guan. 2025b. Quark: Quantization-enabled circuit sharing for transformer acceleration by exploiting common patterns in nonlinear operations. In *2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9. IEEE.

Zhixiong Zhao, Fangxin Liu, Junjie Wang, Chenyang Guan, Zongwu Wang, Li Jiang, and Haibing Guan. 2026. Specquant: Spectral decomposition and adaptive truncation for ultra-low-bit llms quantization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 28786–28794.

Yixiao Zhou, Dongzhou Cheng, Zhiliang Wu, Yi Yang, Yu Cheng, and Hehe Fan. 2026a. One refiner to unlock them all: Inference-time reasoning elicitation via reinforcement query refinement. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 38957–38978.

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

Yixiao Zhou, Ziyu Zhao, Dongzhou Cheng, Zhiliang Wu, Jie Gui, Yi Yang, Fei Wu, Yu Cheng, and Hehe Fan. 2025. Dropping experts, recombining neurons: Retraining-free pruning for sparse mixture-of-experts llms. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 15169–15186.

Jiaru Zou, Soumya Roy, Vinay Kumar Verma, Ziyi Wang, David Wipf, Pan Lu, Sumit Negi, James Zou, and Jingrui He. 2026. TaTToo: Tool-grounded thinking PRM for test-time scaling in tabular reasoning. In *The Fourteenth International Conference on Learning Representations*.

A Additional Experimental Setups

A.1 Model Configurations

In our experiments, we use Qwen3-1.7B (Yang et al., 2025a) as the SRM. For the LRM, we consider two representative settings: Qwen3-14B (Yang et al., 2025a) for same-family collaboration and DeepSeek-R1-Distill-Qwen-14B (Guo et al., 2025) for cross-family collaboration. This design allows us to evaluate whether EcoTab remains effective under both homogeneous and heterogeneous model pairs. For EcoTab and all compared baselines, we use the same decoding configuration for fair comparison. Specifically, we set the temperature to 0.7, the maximum generation length to 16,384 tokens, and top-p sampling with $p = 0.95$.

A.2 Dataset Details

TabFact (Chen et al., 2020a). TabFact is a large-scale benchmark for table-based fact verification rather than standard table question answering. It consists of approximately 16K Wikipedia tables paired with 118K human-annotated natural language statements, where each statement is labeled as either *ENTAILED* or *REFUTED* with respect to the corresponding table. The dataset is challenging because it requires not only semantic understanding of natural language statements, but also symbolic reasoning over semi-structured tables, such as comparison, counting, and aggregation.

WikiTableQuestions (WikiTQ) (Pasupat and Liang, 2015). WikiTQ is a benchmark for answering complex natural language questions over semi-structured HTML tables. It contains 22,033 question-answer pairs over 2,108 Wikipedia tables. A key characteristic of WikiTQ is that the training and test tables are disjoint, which requires models to generalize to unseen table schemas. The questions often involve compositional reasoning, including comparison, superlatives, aggregation,

and arithmetic operations. The tables are semi-structured and non-normalized, and many cells contain multi-part values that must be interpreted appropriately during reasoning.

TableBench (Wu et al., 2025a). TableBench is a comprehensive and challenging benchmark designed to evaluate complex table question answering in more realistic scenarios. It covers 18 fine-grained subcategories under four major categories, namely fact checking, numerical reasoning, data analysis, and visualization, with a total of 886 benchmark instances. The benchmark is built from 3,681 unique tables spanning diverse domains, with an average of 16.71 rows and 6.68 columns per table. In addition, 65.74% of table cells are numerical, and each instance requires 6.26 reasoning steps on average, making TableBench substantially more difficult than earlier TableQA benchmarks.

HiTab (Cheng et al., 2022). HiTab is a benchmark for question answering and natural language generation over hierarchical tables. Unlike prior datasets that mainly focus on flat tables, HiTab emphasizes hierarchical indexing and implicit semantic and numerical relations induced by table structure. It is a cross-domain dataset constructed from statistical reports and Wikipedia pages, and nearly all tables exhibit hierarchical organization. The dataset contains 10,686 QA pairs and descriptive sentences over 3,597 tables, together with fine-grained annotations of entity and quantity alignment, which make it suitable for studying complex reasoning over hierarchical tabular data.

FinQA (Chen et al., 2021). FinQA is a financial-domain dataset for complex numerical reasoning over heterogeneous evidence. It contains 8,281 question-answer pairs annotated by finance professionals, along with gold reasoning programs for explainable evaluation. The dataset is constructed from earnings reports of S&P 500 companies and requires models to integrate information from both tables and accompanying unstructured text. Compared with general-domain TableQA benchmarks, FinQA places greater emphasis on multi-step numerical reasoning and domain-specific financial knowledge.

Validation set in Sec. 4.3. The held-out validation set in Sec. 4.3 is used only for fitting the offline risk mappings and is strictly separated from the final evaluation set. For datasets with an official training split, we construct the validation set solely from the training data. For datasets without a predefined training split, we randomly sample 10% of the original test set as a pseudo-validation split, and use the remaining 90% for final testing. In all cases, the validation data are used only for calibration and are never included in the final reported test results.

A.3 Implementation Details

For the accuracy–FLOPs evaluations in Sec. 3 and Sec. 5, we follow a unified threshold-sweeping protocol for EcoTab and all compared baselines. Specifically, for each method, we perform a grid search over the routing threshold τ with a step size of 0.05, and compute the corresponding accuracy and average FLOPs under different values of τ. This process produces the full accuracy–FLOPs trade-off curve for each method. For the main results in Table 1, we report two representative operating points from the trade-off curve. First, we use the accuracy achieved at 60% of the LRM-only FLOPs as the reported **Acc**. Second, we use the FLOPs required to reach 98% of the LRM-only accuracy as the reported **FLOPs**. These two metrics respectively reflect the model quality under a fixed computation budget and the computation required to approach near-LRM performance. In addition, we report Accuracy-per-FLOPs (A/F) as an overall indicator of the effectiveness–efficiency trade-off (Ma et al., 2025). Following our main experimental setup, FLOPs are estimated using the standard Transformer approximation of $2N$ per generated token for a model with N parameters. All reported results are averaged over three independent runs.

A.4 Four Error Types for Failed Steps

Following TATTOO (Zou et al., 2026), we manually inspect the failed trajectories and categorize each failed step into one of four error types.

Table Retrieval Step. This type includes row or column mis-selection, unit mismatch, and partial aggregation errors. These errors account for 47.7% of all failed steps, indicating that a substantial portion of failures arise from difficulty in correctly locating and extracting the relevant table region.

Table Operation Step. This type covers miscalculation, grouping mistakes, double counting, and misinterpretation of table semantics. It represents 34.3% of all failed steps, suggesting that even after the relevant contents are retrieved, reasoning over structured tabular information remains challenging.

Inner-Thinking Step. This type refers to logical mistakes or self-contradictory reasoning that are

not directly caused by table grounding. Such errors account for 12.0% of all failed steps, indicating that LRMs are relatively more reliable on pure logical chains than on table-centric operations.

Others. This category includes failures caused by context omission, incomplete responses, or improper output formatting.

To provide a more concrete understanding of these error types, Table 3 presents representative failure cases from three major categories: *Table Retrieval*, *Table Operation*, and *Inner-Thinking*. For each case, we show the first erroneous reasoning step in the trajectory and briefly explain how this mistake propagates to the final incorrect answer. These examples illustrate that failures in table reasoning may arise from different stages of the reasoning process, including incorrect table grounding, faulty operations over retrieved contents, and purely logical mistakes.

B Table Trie Construction Details

To separate table tokens from text tokens in each reasoning step, EcoTab builds a word-level Table Trie from the input table. The trie stores normalized textual entries extracted from the table, including column headers and cell values. During inference, each reasoning step is normalized in the same way and scanned from left to right with longest-prefix matching. The matched spans are then mapped back to token positions to obtain the table-token mask used in Eq. (4). This implementation follows the procedure described in Sec. 4.1.

B.1 Normalization Rules

We apply lightweight normalization to both table contents and reasoning steps before trie construction and matching. The goal is to improve robustness to surface-form variation while keeping the procedure simple and efficient.

Lowercasing and whitespace cleanup. All text is converted to lowercase. Consecutive spaces, tabs, and line breaks are collapsed into a single space, and leading or trailing whitespace is removed.

Punctuation normalization. Common punctuation variants are standardized into a unified form. For example, different dashes and quotation marks are mapped to their canonical ASCII forms when possible. Surrounding punctuation that does not affect semantic identity is ignored during matching.

Number normalization. We normalize common number formats to reduce mismatches caused by formatting differences. For example, "1,200" and "1200" are treated as the same value. Decimal numbers and percentages are preserved in normalized form when they carry semantic meaning.

Cell and header insertion. We insert both column headers and cell values into the trie. Each entry is split at the word level after normalization. Multi-word entries such as "new york" or "gross domestic product" are inserted as complete paths rather than as isolated words.

Token-span consistency. Normalization is applied only for matching. After a span is matched in normalized text, we map it back to the original token positions in the reasoning step, so the final boolean mask is still defined over the original generated tokens.

B.2 Trie Matching Algorithm

Given a reasoning step $s_i = (t_{i,1}, t_{i,2}, \ldots, t_{i,k_i})$, we first normalize the step text using the same rules as above. We then scan the step from left to right and perform longest-prefix matching over the trie. At each position, we attempt to extend the current span word by word along the trie. If multiple matches are possible, we keep the longest valid match. For example, if both "new" and "new york" exist in the trie, the matcher prefers "new york" whenever the longer span is observed in the step. Once a match is confirmed, all tokens covered by that span are marked as table-related, and the scan continues from the end of the matched span. If no match is found, the scan advances by one token. Formally, this process returns a boolean mask $\mathbf{m}^{(i)} \in \{0,1\}^{k_i}$, where $\mathbf{m}_j^{(i)} = 1$ indicates that token $t_{i,j}$ belongs to a matched table-related span. Based on this mask, the step tokens are partitioned into the table-token set $V_{\text{tab}}^{(i)}$ and the text-token set $V_{\text{text}}^{(i)}$, which are then used to compute $\Phi_{\text{tab}}^{(i)}$ and $\Phi_{\text{text}}^{(i)}$ in Eq. (4). Although this matching is surface-form based, lightweight normalization already covers manycommon variations in numbers and punctuation, and we find it sufficient in practice.

C Failure-risk Mapping Implementation

C.1 Construction of Fitting Data

EcoTab performs next-step model routing: the routing score computed from the current step s_i is used to decide whether the next step s_{i+1} should be generated by the SRM or the LRM. To align the supervision target with this objective, we construct step-level labels by identifying a critical routing boundary through counterfactual suffix replacement.

Category	Wrong Model Answer	Error Description
Table Retrieval	<think> Step 1: I need to figure out the average prominence of mountain peaks in the Democratic Republic of the Congo that have an elevation of at least 3000 meters. Step 2: First, I'll review the provided table. My main focus is on the 'country' and 'elevation (m)' columns. *(Wrong Step)* . . . </think> Final Answer is \boxed{2306.65}	The model initially retrieves the wrong part of the table (columns *'Country'* and *'Elevation (m)'*), which leads to errors in all subsequent reasoning steps.
Table Operation	<think> Step 1: Let me start by looking at the table provided . . . The retrieved columns are . . . Step 4: I'll go through each row and note down the episodes: Episode 1: Rating - 5.1, Episode 2: Rating - 5.2 . . . *(Wrong Step)* Step 5: Adding the rating together is 5.1 + 5.2 + . . . </think> Final Answer is: \boxed{11.52}	The model identifies the correct column for the answer, but fails to extract all elements, omitting the last cell. This omission leads to incorrect reasoning in all subsequent steps.
Inner-thinking	<think> Step 1: Based on the table contents, I need to count podiums per season. 2008: $(1+1+1)/3 = 1$, 2009: 6, 2010: 0, 2011: $(2+0)/2 = 1$ . . . Step 4: Then I will average across seasons. Final average is $(1+6+0+1)/4 = 2$ *(Wrong Step)* </think> Final Answer is: \boxed{2}	The model incorrectly does the calculation by averaging the season-level means, giving each season equal weight, instead of averaging across all team-seasons.

Table 3: Representative error cases for major failed-step categories.

For each dataset, we first build a held-out validation split following Appendix A.2, and run both LRM-only and SRM-only inference to obtain full reasoning trajectories. We retain only samples that are correct under LRM-only but incorrect under SRM-only, since these cases directly indicate that the stronger model is needed for part of the reasoning process. Each trajectory is segmented into steps using the delimiter "\n\n". For every step s_i in the retained LRM trajectory, we compute the table-token uncertainty $\Phi_{\text{tab}}^{(i)}$ and the text-token uncertainty $\Phi_{\text{text}}^{(i)}$. For a retained trajectory with steps $s_1, \ldots, s_T$, we progressively replace its suffix with SRM generations, starting from the last step. For a suffix length $m \in \{1, \ldots, M\}$ with $M = \min(T-1, 8)$, we keep the prefix $s_1, \ldots, s_{T-m}$ fixed and let the SRM regenerate the remaining suffix. For each m, we repeat generation $k = 5$ times and evaluate the final answer. A suffix is regarded as causing a stable outcome flip if at least 4 out of the 5 runs become incorrect, corresponding to a flip ratio of at least $\gamma = 0.8$.

We then define $m^\star$ as the smallest suffix length that causes a stable outcome flip, and stop the search immediately once it is found. The corresponding critical routing boundary is $b = T - m^\star$. The step s_b is labeled as a positive sample because its routing score should have triggered the switch to the LRM for generating the next step. All earlier steps $s_1, \ldots, s_{b-1}$ are labeled as negative, and all later steps are discarded. If no stable outcome flip is found within the scanned range, the sample is excluded from the fitting set. This construction provides step-level supervision that is better aligned with the next-step routing objective than assigning labels directly from the final trajectory outcome.

Dataset	Retained samples S
WikiTQ	2281
TabFact	2374
TableBench	432
HiTab	1742
FinQA	1165

Table 4: Number of retained step-level samples used for fitting the offline risk mappings under the suffix-replacement construction.

Since the table-token and text-token mappings are fitted on the same retained step set, they share the same binary labels and the same total number of samples. Table 4 reports the total number of retained step-level samples under this construction.

C.2 Fitting Failure-risk Mappings

For each dataset, we fit two independent sigmoid risk mappings, one for table-token uncertainty and the other for text-token uncertainty, using the retained step-level samples constructed above. For each signal $\Phi_*^{(i)}$, where $* \in \{\text{tab}, \text{text}\}$, the risk score is defined as

$$d_*^{(i)} = f_*\left(\Phi_*^{(i)}\right) = \sigma\left(a_* \Phi_*^{(i)} + b_*\right), \quad (7)$$

where $\sigma(\cdot)$ is the sigmoid function, and a_* and b_* are learned from the retained validation samples of that dataset. Let

$$p = \sigma(a_* \Phi + b_*).$$

The fitting objective is the standard binary cross-entropy:

$$\mathcal{L}_* = -\sum_{(\Phi, y) \in \mathcal{D}_*} \left[y \log p + (1-y) \log(1-p) \right]. \quad (8)$$

During inference, EcoTab maps $\Phi_{\text{tab}}^{(i)}$ and $\Phi_{\text{text}}^{(i)}$ into two risk scores and combines them using Noisy-OR:

$$d_{\text{final}}^{(i)} = 1 - \left(1 - d_{\text{tab}}^{(i)}\right)\left(1 - d_{\text{text}}^{(i)}\right). \quad (9)$$

Finally, $d_{\text{final}}^{(i)}$ is compared with the threshold τ to determine whether the next step s_{i+1} should be generated by the LRM or the SRM.

C.3 Discussion

Suffix replacement provides cleaner supervision. Trajectory-level labeling assigns the same positive label to all steps in an incorrect trajectory, even though many early steps may still be handled correctly by the SRM. This introduces label noise and does not align well with next-step routing. In contrast, suffix replacement identifies a critical routing boundary and assigns the positive label only to the step immediately before the shortest suffix whose SRM replacement causes a stable outcome flip. This yields cleaner step-level supervision and requires no additional human annotation.

Offline construction introduces no online overhead. Suffix replacement and risk-mapping fitting are performed only once on the held-out validation set. They are fully offline and do not participate in online inference, so they introduce no additional token generation overhead during routing. At test time, EcoTab only computes uncertainty signals and queries the fitted mappings. As shown in the main paper, EcoTab remains lightweight, with routing overhead comparable to STEER and far lower than RSD, SpecCoT, and SpecReason.

The fitted mapping generalizes across domains. The main paper shows that the learned risk mapping transfers well across domains. In the out-of-domain setting, the mapping fitted on WikiTQ still outperforms Random and STEER when applied directly to TableBench on both Acc@60% LRM-only FLOPs and FLOPs@98% LRM-only Acc. Although the in-domain variant performs slightly better, the gap is small, indicating good cross-domain generalization.

D Additional Related Work

Adaptive Reasoning and Efficient Inference. The reasoning ability of large language models has been continuously extended through both task-specific adaptation and broader application-oriented studies. Recent surveys have summarized LLM evaluation and benchmarking across diverse tasks and scenarios (Chang et al., 2024). Existing work covers optimization and formal reasoning (Liu et al., 2026b; Chen et al., 2024), knowledge-intensive reasoning (Li et al., 2025b, 2026b), software understanding (Zhang et al., 2024), and multimodal tasks (Zhang et al., 2026; Tang et al., 2026a; Xiao et al., 2026a). Similar advances have appeared in LLM agents and retrieval-augmented systems, where models interact with external knowledge sources and perform multi-step reasoning (Liu et al., 2026a; Shi et al., 2026b; Yu et al., 2026a,b; Chen et al., 2026; Wu et al., 2026b; Hao et al., 2026b; Liu et al., 2026d). To strengthen these capabilities, prior studies employ supervised adaptation, parameter-efficient fine-tuning, reinforcement learning, and knowledge augmentation (Chang et al., 2026; Xiao et al., 2026b; Liu et al., 2026c; Lv et al., 2026; Wu et al., 2026a; Li et al., 2026a; Hao et al., 2024, 2026a; Wang et al., 2026). Recent works further investigate adaptive reasoning mechanisms, including structural sensitivity analysis (He et al., 2026), step-wise reasoning optimization (Shi et al., 2026a), and reasoning process manipulation (Liu et al., 2025; Zhou et al., 2026b,a). Alongside capability improvement, reducing inference overhead has become increasingly important. Prior work compresses LLMs through low-bit quantization (Zhao et al., 2025b, 2026; Xu et al., 2026), parameter-efficient adaptation and model optimization (Xu et al., 2025), removes redundant computation in sparse expert architectures (Zhou et al., 2025), or reduces multimodal sequence length through adaptive token compression (Tang et al., 2026b,a). These techniques optimize the execution of a given model from the perspectives of parameters, activations, experts, or input tokens. A complementary direction is to avoid assigning identical computational resources to all inference steps. Model routing addresses this issue by leveraging heterogeneous capability–cost trade-offs among models. Stepwise model routing further extends this idea by dynamically allocating models across intermediate reasoning stages, enabling finer-grained control over the capability–efficiency trade-off.

E Case Study

We further present representative case studies on two key table reasoning skills, namely Table Retrieval and Table Operation, to qualitatively compare SRM-only, STEER, and EcoTab. Figures 10, 11, and 12 show Table Retrieval cases in which both

SRM-only and STEER fail, while EcoTab succeeds. Figures 13, 14, and 15 present analogous cases for Table Operation. The results show that EcoTab can more accurately identify the critical reasoning step and route it to the LRM for handling, leading to correct final predictions.

Case Study on EcoTab (1) Table Retrieval

Question: what is the number of times electrodyne won?

Table: {table content} **Answer:** 5

(i) SRM-only Fail

[Step 1]
Okay, let's see. The question is asking for the number of times Electrodyne won in the 1984 IMSA GT Championship season.

[Step 2]
First, I need to look at the table provided and check the entries where the GTO (Group Touring) winning team is Electrodyne. The table has columns for GTP, GTO, and GTU winning teams and drivers.

[Step 3]
Looking at each row:- Rnd 1: GTO is Stratagraph Inc. - Rnd 2: GTO is Dingman Bros. Racing- Rnd 3: GTO is Stratagraph Inc.- Rnd 4: GTO is Stratagraph Inc.- Rnd 5: GTO is Mandeville Auto Tech- Rnd 6: GTO is Brooks Racing- Rnd 7: GTO is Stratagraph Inc.- Rnd 8: GTO is Mandeville Auto Tech- Rnd 9: GTO is Electrodyne- Rnd 10: GTO is Electrodyne- Rnd 11: GTO is Corvette- Rnd 12: GTO is Brooks Racing- Rnd 13: GTO is Electrodyne- Rnd 14: GTO is Electrodyne- Rnd 15: GTO is Electrodyne- Rnd 16: GTO is Electrodyne- Rnd 17: GTO is Roush Racing

[Step 4] Wrong step

So from Rnd 9 to Rnd 16, there are 6 instances where GTO is Electrodyne. Wait, let me count again: Rnd 9: Electrodyne (1)Rnd 10: Electrodyne (2)Rnd 13: Electrodyne (3)Rnd 14: Electrodyne (4)Rnd 15: Electrodyne (5)Rnd 16: Electrodyne (6)

[Step 5]
Yes, that's six times. So the answer should be 6 ✗

Figure 10: Table Retrieval case with SRM-only.

Case Study on EcoTab (1) Table Retrieval

(ii) STEER Fail

[Step 1] SRM
Okay, let's see. The question is asking for the number of times Electrodyne won in the 1984 IMSA GT Championship season.

[Step 2] SRM
First, I'll look at the structure of the table. Each row represents a race, with columns for Round number, Circuit, and then three categories: GTP, GTO, and GTU winning teams and drivers. The GTO column is where I should focus since Electrodyne is mentioned in that section.

[Step 3] SRM
Looking at Round 3, Sebring: The GTO winner is #4 Stratagraph Inc. So, not Electrodyne.Round 4, Road Atlanta: Again, GTO is #4 Stratagraph Inc.Round 5, Riverside: GTO is #38 Mandeville Auto Tech.Round 6, Laguna Seca: GTO is #77 Brooks Racing.Round 7, Charlotte: GTO is #4 Stratagraph Inc.Round 8, Lime Rock: GTO is #77 Brooks Racing? Wait, no, in Round 8, the GTO team is #38 Mandeville Auto Tech.Wait, looking back at the table, in Round 8, the GTO is #38 Mandeville Auto Tech, so no Electrodyne.Round 9, Mid-Ohio: GTO is #91 Electrodyne! That's one win for Electrodyne.Round 10, Watkins Glen: GTO is #91 Electrodyne again. That's the second win.Round 11, Portland: GTO is #51 Corvette, so no.Round 12, Sears Point: GTO is #77 Brooks Racing.Round 13, Road America: GTO is #91 Electrodyne. That's the third win.Round 14, Pocono: GTO is #65 English Enterprises.Round 15, Michigan: GTO is #84 Dole Racing.Round 16, Watkins Glen: GTO is #91 Electrodyne. That's the fourth win.Round 17, Daytona: GTO is #67 Roush Racing.

[Step 4] SRM critical step is instead routed to the SRM

Let me recount:- Round 9: Electrodyne (1)- Round 10: Electrodyne (2)- Round 13: Electrodyne (3)- Round 16: Electrodyne (4)So that's four times in total.

[Step 5] LRM too late

Wait, but in Round 16, the GTO team is #91 Electrodyne, so that's four wins. Hmm, but I thought maybe only three. Let me double-check each round.Yes, Rounds 9,10,13,16 each have Electrodyne as the GTO winner.

[Step 6] SRM
So that's four times. So the answer should be 4. ✗

Figure 11: Table Retrieval case with STEER.

Case Study on EcoTab (1) Table Retrieval

(iii) EcoTab Correct

[Step 1] SRM
Okay, let's see. The question is asking for the number of times Electrodyne won in the 1984 IMSA GT Championship season.

[Step 2] SRM
First, I need to look at the table provided and check the "GTO Winning Team" column, because that's where Electrodyne is mentioned. The GTO stands for Group Toyota, but in the context of the table, it's probably the teams that won the GTU (Group United) and others. Wait, the columns are GTP, GTO, GTU. The question is about Electrodyne, so I need to check which rows have Electrodyne in the GTO column

[Step 3] SRM
Looking at each row:Rnd 1: GTO is #4 Stratagraph Inc. Not Electrodyne.Rnd 2: GTO is #47 Dingman Bros. Racing. No.Rnd 3: GTO is #4 Stratagraph Inc. No.Rnd 4: GTO is #4 Stratagraph Inc. No.Rnd 5: GTO is #38 Mandeville Auto Tech. No.Rnd 6: GTO is #77 Brooks Racing. No.Rnd 7: GTO is #4 Stratagraph Inc. No.Rnd 8: GTO is #38 Mandeville Auto Tech. No.Rnd 9: GTO is #91 Electrodyne. Yes, here Electrodyne is the winner. So that's one.Rnd 10: GTO is #91 Electrodyne. Another one. So two here.Rnd 11: GTO is #51 Corvette. No.Rnd 12: GTO is #77 Brooks Racing. No.Rnd 13: GTO is #91 Electrodyne. Third one.Rnd 14: GTO is #65 English Enterprises. No.Rnd 15: GTO is #91 Electrodyne. Fourth one.Rnd 16: GTO is #91 Electrodyne. Fifth one.Rnd 17: GTO is #67 Roush Racing. No.

[Step 4] LRM LRM handle the critical step
So counting those instances where GTO Winning Team is #91 Electrodyne, there are 5 times. Let me check again to make sure I didn't miss any.Looking through each row:Rnd 9: Electrodyne (row 9)Rnd 10: Electrodyne (row 10)Rnd 13: Electrodyne (row 13)Rnd 15: Electrodyne (row 15)Rnd 16: Electrodyne (row 16)

[Step 5] SRM
Yes, that's five times. **So the answer should be 5.** ✓

Figure 12: Table Operation case with EcoTab.

Case Study on EcoTab (2) Table Operation

Question: are the dates set in a consecutive order?

Table: {table content} **Answer:** yes

(i) SRM-only Fail

[Step 1]
Okay, let's see. The user is asking if the dates in the table are set in a consecutive order. Let me look at the dates provided.

[Step 2]
First, I need to list out all the dates from the table. Let me check each row:1. 16 May 20102. 13 February 20113. 7 August 20114. 6 November 20115. 20 November 20116. 23 January 20127. 30 January 20128. 17 September 20129. 12 November 201210. 28 October 2013

[Step 3]
Now, I need to check if these dates are in consecutive order. Let's list them in chronological order. Starting from the earliest date: 16 May 2010. Then 13 Feb 2011. Then 7 Aug 2011. Then 6 Nov 2011. Then 20 Nov 2011. Then 23 Jan 2012. Then 30 Jan 2012. Then 17 Sep 2012. Then 12 Nov 2012. Then 28 Oct 2013.

[Step 4] Wrong step
Wait, but between 20 Nov 2011 and 23 Jan 2012, there's a gap. Let me check the dates again. 20 Nov 2011, then 23 Jan 2012. That's a few months later. Then 30 Jan 2012, then 17 Sep 2012, then 12 Nov 2012, then 28 Oct 2013. **So there's a gap between 12 Nov 2012 and 28 Oct 2013. So the dates are not consecutive.**

[Step 5]
Therefore, the answer should be no. ✗

Figure 13: Table Operation case with SRM-only.

Case Study on EcoTab (2) Table Operation

(ii) STEER Fail

[Step 1] SRM
Okay, let's see. The user is asking if the dates in the table are set in a consecutive order. Let me check the dates provided.

[Step 2] SRM
So, first, I'll look at the table provided. The table has multiple entries with dates ranging from May 2010 to October 2013. I need to check the order of these dates to see if they follow one after another without gaps.

[Step 3] SRM
Looking at the dates, I'll list them out in the order they appear:1. 16 May 20102. 13 February 20113. 7 August 20114. 6 November 20115. 20 November 20116. 23 January 20127. 30 January 20128. 17 September 20129. 12 November 201210. 28 October 2013

[Step 4] SRM critical step is instead routed to the SRM
Now, I'll check each consecutive pair to see if each date comes right after the previous one.- 16 May 2010 to 13 February 2011: There's a gap of about 8 months.- 13 February 2011 to 7 August 2011: About 5 months in between.- 7 August 2011 to 6 November 2011: Less than 3 months apart.- 6 November 2011 to 20 November 2011: Only 14 days apart, so consecutive in the same month.- 20 November 2011 to 23 January 2012: About 2 months and 3 days.- 23 January 2012 to 30 January 2012: 7 days apart, same month.- 30 January 2012 to 17 September 2012: 8 months later.- 17 September 2012 to 12 November 2012: About 1 and a half months.- 12 November 2012 to 28 October 2013: **Almost a year gap.**

[Step 5] LRM too late
From this, it's clear that the dates are not in chronological order. For example, after 28 October 2013, the next entry would logically be earlier in 2013, but since that's the last entry, the main issue is that the dates jump around, not following one after another.

[Step 6] SRM ✗
So, the answer is no, they are not in consecutive order.

Figure 14: Table Operation case with STEER.

Case Study on EcoTab (2) Table Operation

(iii) EcoTab Correct

[Step 1] SRM
Okay, let's see. The user is asking if the dates in the table are set in a consecutive order. Let me check the dates provided.

[Step 2] SRM
Looking at the table, the dates are: 16 May 2010, 13 February 2011, 7 August 2011, 6 November 2011, 20 November 2011, 23 January 2012, 30 January 2012, 17 September 2012, 12 November 2012, and 28 October 2013

[Step 3] SRM
I need to check if these dates are in consecutive order. Let's list them out in order:First, sort them by date. Let's see:- 13 Feb 2011- 16 May 2010 (earlier)- 7 Aug 2011- 6 Nov 2011- 20 Nov 2011- 23 Jan 2012- 30 Jan 2012- 17 Sep 2012- 12 Nov 2012- 28 Oct 2013Wait, the first date is 16 May 2010, which is earlier than the others. Then 13 Feb 2011, then 7 Aug 2011, then 6 Nov 2011, 20 Nov 2011, then 23 Jan 2012, 30 Jan 2012, 17 Sep 2012, 12 Nov 2012, and finally 28 Oct 2013

[Step 4] LRM LRM handle the critical step
But the order in the table is starting with 16 May 2010, then 13 Feb 2011, which is correct. Then 7 Aug 2011, then 6 Nov 2011, then 20 Nov 2011, then 23 Jan 2012, 30 Jan 2012, etc. Wait, the dates in the table are listed as they appear in the table. Let me check the original order:The table's dates are:1. 16 May 20102. 13 February 20113. 7 August 20114. 6 November 20115. 20 November 20116. 23 January 20127. 30 January 20128. 17 September 20129. 12 November 201210. 28 October 2013So the dates are in chronological order? **Let me check each consecutive pair**

[Step 5] LRM LRM handle the critical step
From 16 May 2010 to 13 Feb 2011: that's a few months later. Then 7 Aug 2011 is after Feb 2011. Then 6 Nov 2011, then 20 Nov 2011 (which is a few weeks after 6 Nov). Then 23 Jan 2012, then 30 Jan 2012 (a week later). Then 17 Sep 2012, then 12 Nov 2012, then 28 Oct 2013. Wait, 28 October 2013 is after 12 Nov 2012. So the dates are in order from 2010 to 2013, but each entry is after the previous one. **So the dates are in consecutive order.**

[Step 6] SRM ✓
Therefore, the answer is yes, the dates are set in consecutive (chronological) order.

Figure 15: Table Operation case with EcoTab.