TACO: REINFORCING MULTI-TURN TABLE AGENTS WITH HIERARCHICAL CREDIT ASSIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Table agents answer questions over multiple turns, where each turn writes code, executes it, and reads the result. Reinforcement learning on the final answer gives every decision the same credit. Credit assignment promises finer supervision, yet existing methods bring limited gains on table agents. We show that credit should be assigned hierarchically, first across turns and then within each turn, because a turn often chains several table operations of which only some are incorrect. Existing token-level methods, however, hardly distinguish the contributions of different operations within a turn. We further find that hindsight, the information revealed after a turn, shifts the probabilities of its tokens differently for correct and incorrect operations. We therefore propose TACO, a hierarchical Table Credit Optimization framework for multi-turn table agents. Across turns, TACO compares each turn with turns that received the same observations. Within a turn, a hindsight teacher, a periodically synchronized copy of the policy, rescores the turn with its execution feedback and a correct trajectory. The turn credit sets the direction of the update, and the probability gap between the teacher and the policy sets its size for each code and answer token. Extensive experiments on four table reasoning benchmarks across three model scales demonstrate that TACO consistently improves multi-turn table agents and generalizes to benchmarks outside its training data.

1 INTRODUCTION

Table reasoning is a key capability of large language models (LLMs) in real-world applications such as financial analysis (Chen et al., 2021; Zhu et al., 2021) and office assistants (Zhang et al., 2024b; Su et al., 2024). Unlike free text, tables organize information through rows, columns, and cells, requiring models to both understand table structures and perform multi-step reasoning (Zhang et al., 2024c). Recent studies on table reasoning have increasingly shifted from in-context generation (Wang et al., 2024; Ye et al., 2023; Cheng et al., 2023) to interactive agentic reasoning (Lei et al., 2026; Jiang et al., 2025; Zhou et al., 2025). Instead of generating a complete reasoning trajectory at once, table agents interact with tables over multiple turns to perform sub-table retrieval, numerical computation, sorting, revision, and cross-validation, while using environmental feedback to progressively solve complex problems. Recent work further adopts reinforcement learning (RL) to strengthen these multi-turn interaction capabilities (Lei et al., 2026).

Recently, growing data volumes and feature-rich systems have made table reasoning more demanding, requiring longer reasoning trajectories and more complex interactions with tables (Pan et al., 2026; Yang et al., 2026b). This growing complexity calls for stronger multi-turn table agents. Credit assignment offers a promising direction. Instead of letting the whole trajectory share the same reward, it estimates the influence of each part of the trajectory on the final answer and reweights the reward accordingly (Feng et al., 2025b; Wei et al., 2025; Kazemnejad et al., 2024). Such finer-grained supervision has further improved agents in web and embodied tasks such as WebShop and ALFWorld (Feng et al., 2025b; Zhou et al., 2024). This success therefore raises a natural question: **Can credit assignment bring the same improvements to multi-turn table agents?**

To investigate this question, we conduct a series of empirical studies in Section 3. We first revisit several existing credit assignment methods (Feng et al., 2025b; Wei et al., 2025; Tan et al., 2025; Sun et al., 2025) and evaluate their effectiveness in training multi-turn table agents. Our analysis reveals two observations. (1) Existing credit assignment methods provide only limited improvements

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

for agentic table reasoning and fail to consistently outperform vanilla GRPO. (2) Effective credit assignment for table agents requires hierarchical allocation, where credit is first assigned across turns and then further refined within each turn. Further analysis shows that the key to hierarchical credit assignment is to distinguish the contributions of different parts within each turn. Our key insight is that later information, which we call hindsight, can help reveal these differences. Previous studies have shown that incorporating hindsight can change the model's confidence in its earlier decisions (Zhao et al., 2026; Hübotter et al., 2026). We find that these changes provide a useful signal for refining credit within each turn, since they separate the correct and incorrect operations of a turn.

Motivated by these findings, we propose TACO, a hierarchical **Ta**ble **C**redit **O**ptimization framework for multi-turn table agents. TACO assigns credit at two levels. Across turns, it compares each turn with turns that received the same observations in other trajectories (Feng et al., 2025b) to estimate whether the turn improves the final outcome. Within a turn, a hindsight teacher, a periodically synchronized copy of the policy, rescores the turn with its execution feedback and a correct trajectory for the same question. The turn credit sets the direction of the update, and the probability gap between the hindsight teacher and the policy sets its size separately for each code and answer token.

Our Contributions. (1) ***New Perspective***. We show that existing credit assignment methods provide only limited gains for multi-turn table agents. Further analysis reveals that effective credit assignment for table agents requires finer-grained refinement within each turn, while existing token-level methods struggle to distinguish the contributions of different parts within a turn. (2) ***Novel Framework***. We propose TACO, a hierarchical credit assignment framework that first assigns credit across turns based on shared states and then refines credit within each turn using a hindsight teacher. (3) ***Strong Performance***. Extensive experiments on four table reasoning benchmarks across three model scales demonstrate the effectiveness and generalization of TACO, showing that it further strengthens the multi-turn interaction capabilities of table reasoning agents.

2 PRELIMINARY

Agentic Table Reasoning. Given a semi-structured table T and a natural language question q, an agent interacts with an execution environment over multiple turns to derive the final answer. At turn t, the agent observes the current state s_t, produces an action a_t, and receives an observation o_t from the environment. In table reasoning, each action a_t typically involves multiple fine-grained table operations, such as row selection, column selection, filtering, and aggregation. So a trajectory is defined as: $\tau = (s_0, a_0, o_0, \ldots, s_L, a_L, o_L)$. The predicted answer A_{pred} is extracted from the trajectory and compared with the gold answer A_{gold} to obtain a trajectory-level reward $R(\tau)$.

Credit Assignment in Agentic GRPO. Given a trajectory τ and its outcome reward $R(\tau)$, GRPO optimizes the policy by comparing the relative performance among trajectories without training an additional value model (Shao et al., 2024). Specifically, the rewards are normalized across sampled trajectories to compute a trajectory-level advantage:

$$A(\tau) = \frac{R(\tau) - \mu_R}{\sigma_R}. \tag{1}$$

This advantage is then shared across all tokens in the trajectory during policy optimization. However, different parts of a trajectory may have different influences on the final outcome (Feng et al., 2025b; Wei et al., 2025). Credit assignment aims to estimate the influence of each part within a trajectory and provide more fine-grained supervision (Feng et al., 2025b; Ding et al., 2025). Specifically, it assigns a credit score c_i to each part, which reflects its estimated influence on the final outcome. The original trajectory-level advantage can then be reweighted as $A_i = c_i A(\tau)$ for policy optimization.

On-Policy Self-Distillation. On-policy distillation supervises a student on its own samples (Agarwal et al., 2024; Gu et al., 2024). A teacher rescores the tokens the student generated, and the student is updated toward the teacher. In the self-distillation setting, the teacher π_T is a stop-gradient copy of the policy that is periodically synchronized with it, so no external model is needed (Zhao et al., 2026; Hübotter et al., 2026). The quantity it provides is the per-token log-likelihood gap

$$\Delta_{t,i} = \log \pi_T(y_{t,i} \mid s_t, y_{t,<i}) - \log \pi_\theta(y_{t,i} \mid s_t, y_{t,<i}), \tag{2}$$

which measures how much the teacher and the policy disagree on a token the policy already produced. On-policy distillation minimizes this gap as a training loss, whereas we read it as a measure of how much a decision made by the policy agrees with the context the teacher is given.

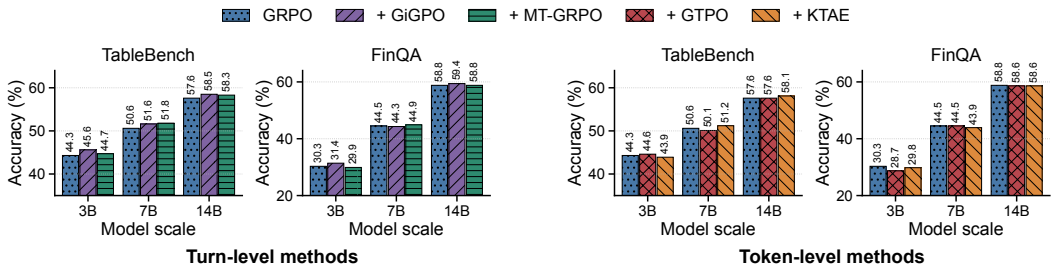

Figure 1: Test accuracy of vanilla GRPO and four credit assignment methods on TableBench and FinQA with Qwen2.5-Coder at 3B, 7B, and 14B, averaged over seeds. Each method changes only how GRPO assigns credit within a trajectory, and all other settings are shared.

3 MOTIVATION

We begin by revisiting existing credit assignment methods to assess their effectiveness for agentic table reasoning. To this end, we conduct a pilot study guided by the following research question.
RQ1 – *For agentic table reasoning, can existing credit assignment methods remain effective and further improve the performance of table agents in multi-turn interactions?*

Effectiveness Analysis. We compare vanilla GRPO (Shao et al., 2024) with four representative credit assignment methods, namely two turn-level methods, GiGPO (Feng et al., 2025b) and MT-GRPO (Wei et al., 2025), and two token-level methods, GTPO (Tan et al., 2025) and KTAE (Sun et al., 2025). We train Qwen2.5-Coder (Hui et al., 2024) at 3B, 7B, and 14B with each method under the same data, reward, and training budget. We evaluate them on TableBench (Wu et al., 2025a) and on FinQA (Chen et al., 2021). Detailed experimental settings are provided in Appendix B. Figure 1 compares the performance of the two types of credit assignment methods with vanilla GRPO. We observe that turn-level methods can provide some improvements, but the overall gains remain limited. In contrast, token-level methods bring fewer benefits and may even degrade performance.
Observation 1. Existing credit assignment methods provide limited improvements for agentic table reasoning and struggle to consistently enhance multi-turn table agents beyond vanilla GRPO.

Error Analysis. To understand why existing credit assignment methods provide limited improvements, we analyze the error patterns of table agents. Specifically, we sample 500 incorrect trajectories shared by all five methods and ask human experts to identify the error that leads to each incorrect answer. We categorize these errors into three types. *Turn-level errors* refer to cases where an entire turn is incorrect. *Intra-turn errors* refer to cases where only part of a turn is incorrect. *Other errors* include the remaining cases, such as formatting mistakes. As shown in Figure 2 (Left), intra-turn errors account for 71% of all errors, while turn-level errors and other errors account for 15% and 14%, respectively. This pattern stems from the composite nature of table agent actions. Unlike web and embodied agents, where a turn usually corresponds to a single atomic operation such as a click or a move (Yao et al., 2022; Shridhar et al., 2021), a table agent turn often consists of multiple table operations, including row selection, column filtering, and sorting. Figure 2 (Right) further shows that incorrect trajectories contain at least two operations in 84% of turns on TableBench and 81% of turns on FinQA. When such a turn fails, only a subset of its operations is typically incorrect. Therefore, assigning a single credit value to the entire turn inevitably mixes correct and incorrect operations, even though they should receive different optimization signals.
Observation 2. Credit assignment for table agents requires hierarchical allocation, where credit is first assigned across the turns of a trajectory and then further refined within each turn.

Why Do Token-Level Methods Provide Limited Gains? Observation 2 highlights the need to refine credit within each turn. Although token-level methods provide a finer granularity, they still bring limited gains in Figure 1. To understand why, we label each token in turns with intra-turn errors according to the correctness of the operation it belongs to. We then compare the average normalized credit assigned by GTPO and KTAE to the two groups. As shown in Figure 3 (Left), both methods provide limited separation between tokens from correct and incorrect operations, indicating that existing token-level methods struggle to distinguish the contribution of different tokens within a turn. Therefore, existing token-level methods struggle to further refine credit within a turn.

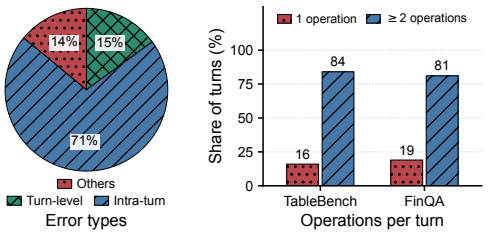

Figure 2: Error analysis of incorrect trajectories. (Left) Share of each error type. (Right) Share of turns with one or at least two operations.

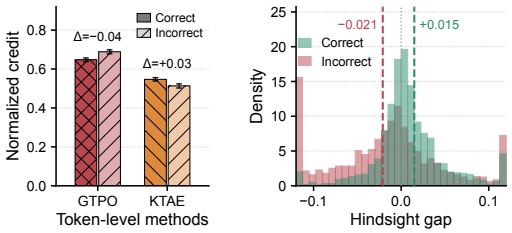

Figure 3: Credit within turns. (Left) Normalized credit and (Right) hindsight gap of correct and incorrect tokens within the same turns.

Motivation for TACO. These findings reveal two requirements for effective credit assignment in table agents. First, credit should be allocated hierarchically, with coarse-grained assignment across turns followed by finer-grained refinement within each turn. Second, intra-turn refinement requires a reliable signal that can capture the contribution of different decisions within a turn.

Our Key Insight. Previous studies have shown that providing models with hindsight information can change the probabilities of previously generated tokens, revealing how the model revises its decisions after observing outcomes (Zhao et al., 2026; Hübotter et al., 2026). Inspired by this observation, we ask whether such probability shifts can serve as a signal to estimate the contribution of different decisions within a turn. To explore this idea, we let the policy revisit each turn together with its feedback and use this hindsight view as a teacher. The probability gap in Eq. (2) measures how the feedback changes the probability of each token. As shown in Figure 3 (Right), within the same turns, the probability gap provides clear separation between tokens from correct and incorrect operations, suggesting that it can serve as a useful signal for intra-turn credit refinement.

4 TACO

Motivated by these findings, we propose TACO, a hindsight-guided credit assignment framework for multi-turn table agents. TACO assigns credit hierarchically, first across turns and then within each turn. It has four stages (Figure 4). It first samples multi-turn trajectories through code execution and scores each trajectory by its final answer (Section 4.1). It then estimates the credit of each turn by comparing it with other turns that received the same observations (Section 4.2). Next, it refines the credit within each turn using hindsight information about that turn (Section 4.3). Finally, it combines the two levels of credit into token-level advantages and updates the policy with them (Section 4.4).

4.1 MULTI-TURN TABLE AGENT ROLLOUT

Credit can only be assigned to decisions that a trajectory makes explicit. We therefore model table reasoning as a multi-turn interaction in which the agent acts on the table through code and observes the result of each execution. We define its state space, action space, and trajectory reward below.

State Space. The table T is loaded as a pandas DataFrame. The prompt x contains the question q, any text that accompanies the table, and the shape, column names, and rows of T, where only the first and last rows are shown for a large table. The state at turn t consists of the prompt and the interaction so far,

$$s_t = (x, a_0, o_0, \ldots, a_{t-1}, o_{t-1}), \tag{3}$$

where a_k and o_k are the action and the observation at turn k. Each code block runs in a fresh process on the original DataFrame, so intermediate results reach later turns only through the observations.

Action Space. An action is a token sequence that consists of a reasoning block followed by either a code block or a final answer,

$$a_t \in \big\{(z_t, c_t),\ (z_t, \hat{y})\big\}, \tag{4}$$

where z_t is the reasoning inside `<think>`, c_t is Python code over the DataFrame inside `<code>`, and $\hat{y}$ is the final answer inside `<answer>`. For a code action, the environment returns an observation o_t that contains the output of c_t, or the error it raises and the line where it occurs. A single code block often filters rows, selects columns, and aggregates values at once, so one action can contain several decisions. An answer action ends the trajectory at turn L. The number of turns is not limited, and a trajectory that fills the context window ends without a final answer.

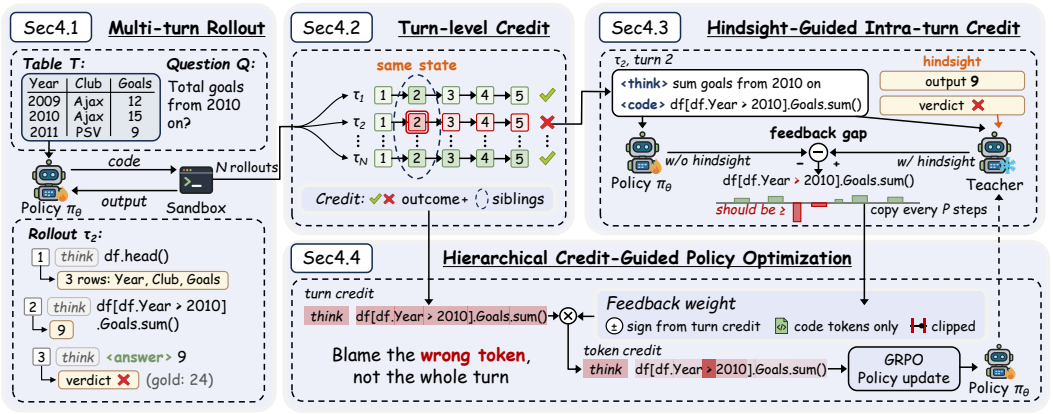

Figure 4: Overview of TACO. The policy samples multi-turn trajectories by writing code and reading its output. Turn-level credit compares turns that share an anchor, and a hindsight teacher that sees the result of each turn reweights its code and answer tokens before the policy update.

Trajectory Reward. When the trajectory ends, the environment compares $\hat{y}$ with the gold answer y^* and assigns the trajectory reward

$$R(\tau) = \begin{cases} 1, & \tau \text{ is valid and } \hat{y} \text{ matches } y^*, \\ 0, & \tau \text{ is valid and } \hat{y} \text{ does not match } y^*, \\ -1, & \tau \text{ is invalid,} \end{cases} \tag{5}$$

where τ is valid if every action follows the required format, the last action is an answer, and at least one code block runs without error. Invalid trajectories thus receive a lower reward than incorrect answers, and the last condition encourages the agent to compute its answer with code rather than read it from the prompt. Numeric answers are matched within a small tolerance and free-form answers with ROUGE-L, and Appendix B gives the prompt and the matching rules for each benchmark.

4.2 Anchor-Guided Turn-Level Credit

The trajectory advantage $A(\tau)$ gives every turn in a trajectory the same value, although an incorrect trajectory may contain useful turns and a correct trajectory may contain unnecessary ones. To measure the contribution of a single turn, we compare it with other turns that received the same observations. All trajectories of a question start from the same prompt, and code execution is deterministic, so different trajectories can receive the same observations and then take different actions.

We therefore group turns by anchors, following GiGPO (Feng et al., 2025b). The anchor of turn t is $\bar{s}_t = (x, o_0, \ldots, o_{t-1})$, the state of Eq. (3) without the earlier actions. Turns with the same anchor hold the same information about the table, although they may have obtained it with different code, so the relative term below compares different actions taken with the same information. Let $\mathcal{U}(t)$ denote the set of turns among the N trajectories sampled for the question whose anchor equals $\bar{s}_t$, including turn t itself. Each turn receives a discounted outcome score $g_t = \gamma^{L-t} R(\tau)$. We normalize this score within $\mathcal{U}(t)$ and add the result to the trajectory advantage,

$$A_t = A(\tau) + \omega \frac{g_t - \text{mean}_{\mathcal{U}(t)}(g)}{\text{std}_{\mathcal{U}(t)}(g)}, \tag{6}$$

where γ is a discount factor and ω weights the relative term. The relative term is set to 0 when all scores in $\mathcal{U}(t)$ are equal. The trajectory advantage keeps the final outcome in every turn, while the relative term separates the turns that lead to better outcomes from the other turns with the same anchor. This term reuses the sampled trajectories, so it requires neither extra rollouts nor a value model. However, all tokens in a turn share A_t, so this stage cannot separate the decisions within it.

4.3 Hindsight-Guided Intra-Turn Credit

Information that becomes available after a turn can guide the credit within that turn. We extract this information with a hindsight teacher π_T, a copy of the policy that receives no gradients and is synchronized with π_θ every P optimizer steps, following on-policy self-distillation (Zhao et al.,

2026; Hübotter et al., 2026). The teacher does not generate actions. Instead, it re-scores each turn with a hindsight context Φ_t placed before the action. The hindsight context can hold any information that the policy cannot see when it acts, such as the execution feedback of the turn, the gold answer, or a correct trajectory for the same question. By default, Φ_t contains the execution feedback of turn t and a correct trajectory from the same group when the group contains one. The execution feedback of a code turn is the result of c_t, namely whether the code runs, what it outputs, or which error it raises and on which line, and that of the answer turn is the verdict of the answer checker together with the gold answer. Section 5.4 compares other choices of Φ_t. The hindsight context is wrapped in a marker that only the teacher sees, so it never enters the input of the policy.

With this context, the probability gap of Eq. (2) becomes

$$\Delta_{t,i} = \log \pi_T\big(y_{t,i} \mid s_t, \Phi_t, y_{t,<i}\big) - \log \pi_\theta\big(y_{t,i} \mid s_t, y_{t,<i}\big), \tag{7}$$

where $y_{t,i}$ is the i-th token of action a_t. A positive gap means that the hindsight context makes the token more likely, and a negative gap means that it makes the token less likely. Tokens in the same turn can receive gaps of opposite signs, which a single credit per turn cannot represent.

The gap measures how strongly the hindsight context supports each token, whereas A_t measures whether the turn improves the final outcome. We therefore let A_t decide the direction of the update and let $\Delta_{t,i}$ adjust the size of the credit of each token,

$$w_{t,i} = \mathrm{clip}\Big(\exp\big(\mathrm{sg}[\mathrm{sgn}(A_t) \cdot \Delta_{t,i}/\kappa]\big), 1-\varepsilon_w, 1+\varepsilon_w\Big), \tag{8}$$

where $\mathrm{sg}[\cdot]$ stops the gradient, κ is a temperature, and ε_w bounds the weight. In a turn with positive credit, tokens supported by the hindsight context receive larger weights and contradicted tokens receive smaller ones. Negative credit reverses this direction. Since the weight is positive and bounded, it keeps the sign of the turn credit, and no single token can dominate the update.

The execution feedback describes the executed code and the checked answer rather than the reasoning around them. We therefore apply the weight only to the code tokens c_t and the answer tokens $\hat{y}$ through a mask $m_{t,i} \in \{0, 1\}$,

$$\bar{w}_{t,i} = (1 - m_{t,i}) + m_{t,i}\, w_{t,i}, \tag{9}$$

so the reasoning tokens z_t keep the original turn credit A_t, and their probability gaps are left unused.

4.4 Hierarchical Credit-Guided Policy Optimization

We combine the turn-level credit and the intra-turn weight into a token-level advantage,

$$\hat{A}_{t,i} = A_t\big[(1-\alpha) + \alpha\, \bar{w}_{t,i}\big], \tag{10}$$

where $\alpha \in [0, 1]$ controls the strength of the intra-turn refinement. With $\alpha = 0$, the advantage reduces to the turn-level credit of Section 4.2. We decay α linearly from 1 to 0 over the first part of training, so the refinement acts most strongly early and later updates rely on the turn credit alone.

The policy is optimized with the clipped GRPO objective, in which $\hat{A}_{t,i}$ replaces the trajectory advantage $A(\tau)$,

$$\mathcal{J}(\theta) = \frac{1}{Z} \sum_{n=1}^{N} \sum_{t} \sum_{i=1}^{|a_t|} \Big[\min\big(r_{t,i}\hat{A}_{t,i},\, \mathrm{clip}(r_{t,i}, 1-\epsilon, 1+\epsilon)\hat{A}_{t,i}\big) - \beta\, \mathbb{D}_{\mathrm{KL}}\big(\pi_\theta \,\|\, \pi_{\mathrm{ref}}\big)\Big], \tag{11}$$

where $r_{t,i}$ is the probability ratio of token $y_{t,i}$ between π_θ and the sampling policy, π_{ref} is the initial policy, β weights the KL term, and Z is the number of action tokens in the group. The sums run over action tokens only, since the observations are produced by the environment.

In each training step, TACO samples groups of trajectories, computes the turn-level credits with Eq. (6), obtains the probability gaps with Eq. (7), and updates the policy with Eq. (11). Compared with GRPO, TACO adds no trainable parameters and no extra rollouts, and its only extra cost is one teacher forward pass without gradients. This design follows the hierarchy of Observation 2, since the turn-level credit decides which turns to reinforce and the intra-turn weight decides how that credit is distributed within each turn. At inference time, the policy runs without the teacher, so its serving cost is the same as that of an agent of the same size trained with GRPO.

Table 1: Main results of TACO and the baselines on four different table reasoning benchmarks. We report results at three model scales, namely 3B, 7B, and 14B. Among the trained methods of each scale, the best result is in **bold** and the second best result is underlined.

Method	Qwen2.5-Coder-3B-Instruct					Qwen2.5-Coder-7B-Instruct					Qwen2.5-Coder-14B-Instruct				
	TB	WTQ	FinQA	HiTab	Avg.	TB	WTQ	FinQA	HiTab	Avg.	TB	WTQ	FinQA	HiTab	Avg.
Base	31.0	38.5	10.4	24.3	26.1	36.8	51.8	40.4	39.2	42.0	46.7	62.8	54.5	51.8	54.0
TaREx	42.2	59.2	28.9	40.7	42.8	49.1	68.3	43.0	47.3	51.9	55.3	75.2	57.4	56.3	61.1
GRPO	44.3	60.1	30.3	41.4	44.0	50.6	69.0	44.5	50.8	53.7	57.6	75.3	58.8	56.9	62.2
+ GiGPO	45.6	60.3	**31.4**	43.5	45.2	51.6	70.1	44.3	51.8	54.5	58.5	76.1	**59.4**	58.1	63.0
+ MT-GRPO	44.7	59.3	29.9	44.9	44.7	51.8	69.7	44.9	**52.7**	54.8	58.3	76.9	58.8	57.6	62.9
+ GTPO	44.6	56.3	28.7	41.9	42.9	50.1	69.0	44.5	51.0	53.7	57.6	75.5	58.6	56.6	62.1
+ KTAE	43.9	58.2	29.8	43.7	43.9	51.2	69.3	43.9	51.1	53.9	58.1	76.3	58.6	56.8	62.5
+ TACO (Ours)	**46.7**	**62.1**	31.1	**45.9**	**46.5**	**53.3**	**71.1**	**45.8**	52.5	**55.7**	**60.2**	**78.3**	**59.4**	**60.8**	**64.7**

5 EXPERIMENTS

In this section, we evaluate the effectiveness of TACO. We first introduce the experimental setup (Section 5.1). We then organize our experiments around four research questions. **Q1**: How does TACO compare with existing credit assignment methods across different model scales? **Q2**: How does each component contribute to the performance of TACO? **Q3**: How does the choice of hindsight context affect TACO? **Q4**: How does TACO compare with existing single-turn table reasoning methods, which answer the question in a single response without executing code?

5.1 EXPERIMENTAL SETUP

Baselines and Models. We compare TACO with both multi-turn and single-turn table reasoning methods. For multi-turn reasoning, we include the untrained agent (Base), TaREx (Lei et al., 2026), GRPO (Shao et al., 2024), and the four credit assignment methods of Section 3, namely GiGPO (Feng et al., 2025b) and MT-GRPO (Wei et al., 2025) at the turn level and GTPO (Tan et al., 2025) and KTAE (Sun et al., 2025) at the token level. The credit assignment methods follow the setup of Section 3 and change only how GRPO assigns credit. For single-turn reasoning, we include the base model, SFT following Table-R1 (Yang et al., 2025), and RL following Table-R1 (Wu et al., 2025b), all of which answer without executing code. Regarding the policy models, we train Qwen2.5-Coder-Instruct (Hui et al., 2024) at 3B, 7B, and 14B. Appendix B.3 details the baselines and models.

Benchmarks and Metrics. We evaluate on four table reasoning benchmarks, namely TableBench (TB) (Wu et al., 2025a) with fact checking, numerical reasoning, and data analysis questions, WikiTQ (WTQ) (Pasupat & Liang, 2015) with questions over Wikipedia tables, FinQA (Chen et al., 2021) with numerical questions over financial reports, and HiTab (Cheng et al., 2022) with questions over hierarchical tables. All methods are trained on the training splits of TableBench and WikiTQ, so FinQA and HiTab are out-of-domain. We report accuracy averaged over training seeds, and Appendix B.1 describes the data and the answer matching rules in detail.

Implementation Details. We implement TACO and all baselines in one codebase with full-parameter training. All training and inference experiments are conducted on 6×H800-80G GPUs. Additional implementation details, including the hyperparameters of all methods and the validation sets used to select the checkpoint of each run, are provided in Appendix B.

5.2 MAIN RESULTS (Q1)

Table 1 compares TACO with the multi-turn baselines at three model scales. TACO reaches the highest average accuracy at every scale and improves over GRPO on all four benchmarks. The existing credit assignment methods stay close to GRPO, as observed in Section 3. The turn-level methods GiGPO and MT-GRPO gain about one point on average, while the token-level methods GTPO and KTAE barely differ from GRPO at 7B and 14B and fall below it at 3B. TACO gains one to two more points over the strongest of these methods at all three scales. This ordering matches the two requirements derived in Section 3. A turn-level credit separates useful turns from harmful ones but gives all tokens inside a turn the same signal, and a token-level credit refines the signal inside a turn but hardly separates correct tokens from incorrect ones. TACO refines the turn-level credit with the hindsight signal and thus meets both requirements. The gains also appear on FinQA and HiTab, which are out-of-domain, so the finer credit improves the agent beyond its training questions.

Table 2: Ablation study on the 7B model. Each row removes one component of TACO.

Variant	TB	WTQ	FinQA	HiTab	Avg.
w/o turn-level credit	52.0	69.9	44.3	51.5	54.4
w/o intra-turn credit	51.6	70.1	44.3	51.8	54.5
w/o token mask	52.8	70.2	45.1	51.7	55.0
w/o weight clip	52.4	70.6	45.4	52.1	55.1
w/o α decay	52.9	71.1	45.3	52.0	55.3
TACO (Full)	**53.3**	**71.1**	**45.8**	**52.5**	**55.7**

Table 3: Hindsight context study on the 7B model. The last row is the default of TACO.

Hindsight context	TB	WTQ	FinQA	HiTab	Avg.
Output only	52.2	70.5	44.9	51.7	54.8
Immediate feedback	52.8	70.8	45.3	52.0	55.2
Full feedback	52.3	70.9	44.9	51.9	55.0
Correct traj. only	52.6	70.9	**45.8**	51.7	55.3
Full + correct traj.	53.0	**71.3**	45.6	52.3	55.6
Immediate + correct traj.	**53.3**	71.1	**45.8**	**52.5**	**55.7**

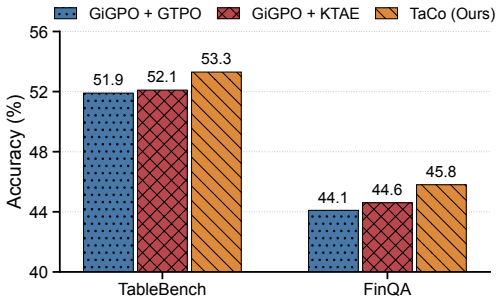

Figure 5: GiGPO combined with token-level methods on the 7B model. All three methods keep the turn-level credit of GiGPO.

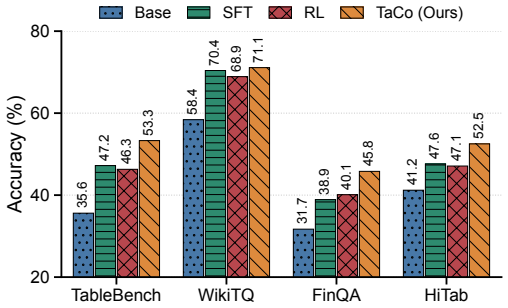

Figure 6: Comparison with single-turn methods on the 7B model. Base, SFT, and RL answer in a single turn without executing code.

5.3 Ablation Study (Q2)

Component Ablation. Table 2 removes one component of TACO at a time on the 7B model. Removing the turn-level credit replaces A_t with $A(\tau)$, removing the intra-turn credit sets every weight to one and leaves GiGPO, and the other rows apply the weight to every token, leave it unbounded, or keep $\alpha = 1$. Every component contributes, and the two stages contribute the most. Removing the turn-level credit costs the most, since the weight then follows the sign of $A(\tau)$ and also reinforces harmful turns in correct trajectories. The drop is largest on FinQA, whose questions take the most turns, so the turn credit matters most when a trajectory has many turns to tell apart. Removing the intra-turn credit costs almost as much, in line with Section 3, where most errors lie inside a turn. Here the drop is largest on TableBench, which has the most composite turns in Figure 2, so each stage matters most where the errors it targets are most frequent. Among the design choices of the weight, the token mask matters most, because the execution feedback describes the code and the answer but not the reasoning, so the gaps of reasoning tokens carry little signal.

Combining GiGPO with Token-Level Methods. A simpler hierarchy stacks an existing token-level method on a turn-level one. We keep the turn-level credit of GiGPO and replace the hindsight weight with the token credit of GTPO or KTAE. GTPO weights each token by the entropy of the policy, and KTAE by how much more often the token appears in correct rollouts than in incorrect ones. As Figure 5 shows, both combinations stay within half a point of GiGPO on TableBench and FinQA, while TACO gains 1.7 and 1.5 points over it. The two-stage structure alone is therefore not enough. The token credits of GTPO and KTAE hardly separate correct tokens from incorrect ones (Section 3), whereas the hindsight gap is derived from the execution feedback of the turn, so it gives the intra-turn weight a more informative signal. KTAE comes slightly closer to TACO than GTPO on both benchmarks, as its statistics depend on the outcome and the entropy of GTPO does not.

5.4 In-Depth Analysis on TACO (Q3 & Q4)

Effect of Hindsight Context. Table 3 trains TACO on the 7B model with different hindsight contexts Φ_t. Output only gives the printed result or error of the code and, for the answer turn, the gold answer. Immediate feedback adds whether the code ran and whether the answer is correct, and full feedback gives this feedback for every turn. Correct traj. only gives a correct trajectory from the same group, and the last row adds it to the full feedback. Every alternative lowers the average accuracy. Without the correct trajectory, the immediate feedback still reaches 55.2, above every 7B baseline in Table 1 and 0.5 points below the default. The status and the verdict help the teacher read the output, while the feedback of other turns hurts, since it describes code that the scored turn does

not contain. The feedback of the turn and the correct trajectory are complementary, because the first tells whether the turn worked and the second shows how the question can be solved. A correct trajectory alone matches the default on FinQA, where a question reduces to one chain of arithmetic that the trajectory spells out, while the immediate feedback helps more on TableBench and HiTab, where the printed result of a turn shows whether it selected the right rows and columns.

Comparison with Single-Turn Methods. Figure 6 compares TACO with single-turn methods on the four benchmarks at 7B. Base prompts the untrained model to answer from the table in one response. SFT and RL follow the two Table-R1 methods of Yang et al. (2025) and Wu et al. (2025b), respectively, and we retrain both on the same questions as TACO. TACO outperforms both trained single-turn methods on every benchmark. The gain is larger on TableBench and FinQA, where the multi-turn GRPO agent takes 3.3 and 3.7 turns per question, than on WikiTQ and HiTab, where it takes 2.6 and 2.9. SFT and RL stay within 1.5 points of each other on every benchmark, so the choice of single-turn recipe matters little. Executing code and reading its result thus improves accuracy beyond what single-turn training provides, more so for questions that need more turns.

6 Related Work

Table reasoning with LLMs. Early work encodes tables with pre-trained encoders (Herzig et al., 2020; Yin et al., 2020). With LLMs, prompting methods decompose the question or transform the table step by step (Ye et al., 2023; Wang et al., 2024; Jiang et al., 2023), or delegate the computation to programs (Cheng et al., 2023; Chen et al., 2023; Gao et al., 2023). Instruction tuning then builds table-specialized models (Zhang et al., 2024a; Zha et al., 2023; Su et al., 2024; Zhang et al., 2024b), and RL with verifiable rewards trains models to reason before they answer in a single pass (Wu et al., 2025b; Yang et al., 2025; Lei et al., 2025). More recent table agents solve a question over several turns, in which they write code, read its execution result, and decide the next step (Lei et al., 2026; Jiang et al., 2025; Zhou et al., 2025; Kwok et al., 2026). TaREx (Lei et al., 2026) trains such an agent with GRPO on the final answer, so all turns and tokens of a trajectory share the same advantage. We follow this setting and study how to distribute the outcome reward over turns and tokens.

Credit assignment in multi-turn RL. RL for multi-turn tool-use agents usually assigns one advantage to the whole trajectory (Feng et al., 2025a; Jin et al., 2025; Xue et al., 2025; Wang et al., 2025). Credit assignment methods refine this signal at the turn level or at the token level. Turn-level methods give each turn its own credit. GiGPO (Feng et al., 2025b) compares actions taken from the same state across trajectories, MT-GRPO (Wei et al., 2025) and Ding et al. (2025) add turn-level rewards, and ArCHer (Zhou et al., 2024) and VinePPO (Kazemnejad et al., 2024) estimate values with a learned critic or with Monte Carlo rollouts. However, all tokens within a turn still share one credit. Token-level methods assign credit to individual tokens. GTPO (Tan et al., 2025) shapes token rewards with policy entropy, KTAE (Sun et al., 2025) scores key tokens by comparing correct and incorrect rollouts, and PRIME (Cui et al., 2025) obtains token rewards from an implicit process reward model. On-policy self-distillation derives token credit from a copy of the policy that sees privileged information (Zhao et al., 2026; Hübotter et al., 2026). None of these methods is designed for table agents, where a single turn often contains several decisions. TACO assigns credit to such agents in two stages. It first compares turns that share an anchor to obtain turn-level credit, and then refines the credit within each turn with hindsight about that turn, such as its execution result.

7 Conclusion

We studied credit assignment for multi-turn table reasoning agents trained with RL. Trajectory-level rewards give every step the same signal, and we showed that this signal is wrong on many individual steps. TACO fixes this with the execution feedback that every step already produces and a correct trajectory for the same question. A turn credit computed from anchors shared across trajectories decides which turns to reinforce, and a hindsight teacher scores each turn with its own feedback and this trajectory, so that the gap to the policy shares this credit among the tokens of the turn. The turn credit sets the direction of the update and the hindsight sets its size per token, at the cost of one extra forward pass and no labels. On four benchmarks and three model scales TACO improves over multi-turn GRPO and makes the agent's answers more traceable to executed results. Two directions follow. The feedback gap could steer sampling toward the steps it identifies as weak, and the same rule could extend to multi-table questions and to spreadsheet editing, where every step still returns a computed result that the teacher can read, such as a joined table or an edited cell.

AI USE STATEMENT

Generative AI assisted manuscript revision, plotting code, and checks of reported aggregates against stored results, and also served as the base models in our training experiments.

ETHICS STATEMENT

This work trains and evaluates table agents on the public benchmarks TableBench, WikiTQ, FinQA, and HiTab under their licenses, and it collects no data from people. The agents run the Python code they generate, so we execute every code block in a separate process with limits on time and memory. Agents deployed on real tables should run their code in a comparable sandbox, and their answers on financial reports must not be interpreted as financial or investment advice.

REPRODUCIBILITY STATEMENT

Upon acceptance, we will publicly release the code, data, and analysis artifacts, including all necessary resources for reproducing our experiments. Appendix B describes the training data, the test sets, the answer matching rules, the hyperparameters of all methods, the full prompts, and the construction of the hindsight context, and Appendix C gives the protocols of the pilot study.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations (ICLR)*, 2024.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research (TMLR)*, 2023.

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. Finqa: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 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 (ACL)*, 2022.

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. Binding language models in symbolic languages. In *International Conference on Learning Representations (ICLR)*, 2023.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Yifeng Ding et al. Empowering multi-turn tool-integrated agentic reasoning with group turn policy optimization. *arXiv preprint arXiv:2511.14846*, 2025.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in LLMs. *arXiv preprint arXiv:2504.11536*, 2025a.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. *arXiv preprint arXiv:2505.10978*, 2025b.

Mingkuan Feng, Jinyang Wu, Hao Gu, Fangrui Lv, Ruihan Jin, Chuyuan Zhang, Zhengqi Wen, and Jianhua Tao. TACO: Tool-augmented credit optimization for agentic tool use. *arXiv preprint arXiv:2606.30251*, 2026.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *International Conference on Machine Learning (ICML)*, 2023.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 32694–32717, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/8ac015d409635f196f9e3e9dcfb9a94e-Paper-Conference.pdf.

Shuo He, Lang Feng, Qi Wei, Xin Cheng, Lei Feng, and Bo An. Hierarchy-of-groups policy optimization for long-horizon agentic tasks. In *International Conference on Learning Representations (ICLR)*, 2026.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2.5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Chuang Jiang, Mingyue Cheng, Xiaoyu Tao, Qingyang Mao, Jie Ouyang, and Qi Liu. Tablemind: An autonomous programmatic agent for tool-augmented table reasoning. *arXiv preprint arXiv:2509.06278*, 2025.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. Structgpt: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. *arXiv preprint arXiv:2410.01679*, 2024.

Tung Sum Thomas Kwok et al. TabQAWorld: Optimizing multimodal reasoning for multi-turn table question answering. *arXiv preprint arXiv:2604.03393*, 2026.

Fangyu Lei, Jinxiang Meng, Yiming Huang, Tinghong Chen, Yun Zhang, Shizhu He, Jun Zhao, and Kang Liu. Reasoning-table: Exploring reinforcement learning for table reasoning. *arXiv preprint arXiv:2506.01710*, 2025.

Fangyu Lei, Jinxiang Meng, Yiming Huang, Shizhu He, Jun Zhao, and Kang Liu. TaREx: Reinforcement learning for code-driven table reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 31501–31509, 2026.

Changzai Pan et al. ReasonTabQA: A comprehensive benchmark for table question answering from real world industrial scenarios. *arXiv preprint arXiv:2601.07280*, 2026.

Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2015.

Jiangweizhi Peng, Yuanxin Liu, Ruida Zhou, Charles Fleming, Zhaoran Wang, Alfredo Garcia, and Mingyi Hong. HiPER: Hierarchical reinforcement learning with explicit credit assignment for large language model agents. *arXiv preprint arXiv:2602.16165*, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Yang Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations (ICLR)*, 2021.

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

Wei Sun, Wen Yang, Pu Jian, Qianlong Du, Fuwei Cui, Shuo Ren, and Jiajun Zhang. KTAE: A model-free algorithm to key-tokens advantage estimation in mathematical reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Hongze Tan, Zihan Wang, Jianfei Pan, Jinghao Lin, Hao Wang, Yifan Wu, Tao Chen, Zhihang Zheng, Zhihao Tang, and Haihua Yang. GTPO and GRPO-S: Token and sequence-level reward shaping with policy entropy. *arXiv preprint arXiv:2508.04349*, 2025.

Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. Hindsight credit assignment for long-horizon llm agents. *arXiv preprint arXiv:2603.08754*, 2026.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

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. Chain-of-table: Evolving tables in the reasoning chain for table understanding. In *International Conference on Learning Representations (ICLR)*, 2024.

Quan Wei et al. Reinforcing multi-turn reasoning in LLM agents via fine-grained reward structure and credit assignment. *arXiv preprint arXiv:2505.11821*, 2025.

Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xeron Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, Tongliang Li, Zhoujun Li, and Guanglin Niu. Tablebench: A comprehensive and complex benchmark for table question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25497–25506, 2025a.

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

Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. Simpletir: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. *arXiv preprint arXiv:2509.02479*, 2025.

Haojin Yang, Ai Jian, Yiwei Wang, Xinyue Huang, Weipeng Zhang, Ke Zeng, Xunliang Cai, and Jingqing Ruan. Harmonizing dense and sparse signals in multi-turn RL: Dual-horizon credit assignment for industrial sales agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pp. 1067–1076, 2026a.

Zhen Yang, Wei Du, Jie Wang, Wenze Zhou, Xiangfeng Meng, Zhengyang Wang, Suping Sun, Ziwei Du, Haodong Zou, Jie Chen, Yongbin Liu, Shicheng Tan, Jiahao Ying, and Shu Zhao. CompTab: A comprehensive benchmark for real-world TableQA with complex reasoning and irregular tables. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27761–27774, 2026b.

Zheyuan Yang, Lyuhao Chen, Arman Cohan, and Yilun Zhao. Table-r1: Inference-time scaling for table reasoning. *arXiv preprint arXiv:2505.23621*, 2025.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 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*, 2023.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

Liangyu Zha, Junlin Zhou, Liyao Li, Rui Wang, Qingyi Huang, Saisai Yang, Jing Yuan, Changbao Su, Xiang Li, Aofeng Su, et al. Tablegpt: Towards unifying tables, nature language and commands into one GPT. *arXiv preprint arXiv:2307.08674*, 2023.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 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 (NAACL)*, 2024a.

Xiaokang Zhang, Sijia Luo, Bohan Zhang, Zeyao Ma, Jing Zhang, Yang Li, Guanlin Li, Zijun Yao, Kangli Xu, Jinchang Zhou, et al. Tablellm: Enabling tabular data manipulation by LLMs in real office usage scenarios. *arXiv preprint arXiv:2403.19318*, 2024b.

Xuanliang Zhang, Dingzirui Wang, Longxu Dou, Qingfu Zhu, and Wanxiang Che. A survey of table reasoning with large language models. *arXiv preprint arXiv:2402.08259*, 2024c.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn RL. In *International Conference on Machine Learning (ICML)*, 2024.

Yuhang Zhou, Mingrui Zhang, Ke Li, Mingyi Wang, Qiao Liu, Qifei Wang, Jiayi Liu, Fei Liu, Serena Li, Weiwei Li, et al. Mixture-of-minds: Multi-agent reinforcement learning for table understanding. *arXiv preprint arXiv:2510.20176*, 2025.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

CONTENTS OF THE APPENDIX

A Additional Related Work **15**

B Additional Implementation Details **15**

B.1 Data and Evaluation 15

B.2 Rewards and Advantages 16

B.3 Policy Models and Multi-Turn Baselines 16

B.4 Combinations of GiGPO and Token-Level Methods 17

B.5 Single-Turn Baselines 17

B.6 Training and Hyperparameters 17

B.7 Execution Environment 17

B.8 Prompts and Action Format 17

B.9 Hindsight Context 20

C Additional Details of the Pilot Study **21**

C.1 Settings 21

C.2 Error Annotation 21

C.3 Operation Counting 21

C.4 Token Credit of GTPO and KTAE 21

C.5 Hindsight Gap 21

D Additional Experimental Details **21**

A Additional Related Work

Hierarchical Credit Assignment. Several recent methods assign credit at more than one level of a long interaction. HiPER (Peng et al., 2026) splits the policy into a planner that proposes subgoals and an executor that carries them out, and it estimates advantages at both levels. DuCA (Yang et al., 2026a) trains sales dialogue agents with turn-level and session-level rewards and normalizes the advantages of the two horizons separately before combining them. HGPO (He et al., 2026) extends the step grouping of GiGPO to a hierarchy of groups whose members share histories of different lengths, and it combines the advantages computed in these groups. HCAPO (Tan et al., 2026) uses the LLM itself as a critic that revises the value of each step with hindsight reasoning after the outcome is known. In all of these methods, the finest unit of credit is a step or a turn, so the tokens of one turn still share one value. TACO adds a level below the turn, at which the tokens of one turn receive different weights derived from the scores of the hindsight teacher.

Credit for Tool Calls. The TACO method of Feng et al. (2026), which shares only its name with TACO, trains multimodal agents that call code tools on images. It credits each tool call by the change in the outcome reward when the model answers with and without the call. Its unit of credit is a whole tool call, whereas TACO distributes the credit of one turn among its tokens.

B Additional Implementation Details

B.1 Data and Evaluation

Training Data. Training uses questions from TableInstruct, the training set released with TableBench (Wu et al., 2025a), and from the training split of WikiTQ (Pasupat & Liang, 2015). We leave out the HiTab file of TableInstruct, so that HiTab stays out-of-domain. The candidate pool contains 3,682 TableInstruct questions and 11,256 WikiTQ questions. We then filter it by how difficult each question is for the untrained agent. The agent built on Qwen2.5-Coder-3B-Instruct samples eight trajectories per question at the training temperature, with the prompt, environment, and reward of Section 4.1. A question solved by all eight trajectories is removed, since its group gives zero advantage to every trajectory. A question solved by none of them is kept unless its label is wrong, because many such questions become solvable during training. For these questions, Qwen3-14B checks whether the gold answer can be derived from the table, and we remove a question only when a majority of its votes reject the label. After the filter, we keep all 1,881 TableInstruct questions and sample 3,762 WikiTQ questions with a fixed seed, so WikiTQ has twice as many questions as TableInstruct. The final pool has 5,643 questions, none of which appears in a test or validation set.

Each table is loaded as a pandas DataFrame whose columns follow the header of the table. A table with a hierarchical header is flattened by joining the header levels of each column into one name.

Test Sets. Table 4 summarizes the four test sets. We remove the visualization questions of TableBench, whose answers are charts rather than values, and the few test questions whose prompts do not fit the context window. The prompts of FinQA contain the report text around the table, since many of its answers need a number from the text, and the prompts of HiTab contain the table caption. HiTab has no split that is disjoint from TableInstruct, so we hold out 2,000 questions from the HiTab file of TableInstruct, which never enters training, and keep the 1,998 that fit the context window. Our HiTab numbers are therefore not comparable to results on the official HiTab test set.

Table 4: Test sets. FinQA and HiTab are out-of-domain, since no training question comes from them. The last column summarizes the answer matching rule of each benchmark.

Benchmark	Questions	Domain	Answer matching
TableBench	829	in-domain	by subtype, following the official rule
WikiTQ	4,296	in-domain	normalized match, lists compared as sets
FinQA	1,129	out-of-domain	numeric match, also in percent scale
HiTab	1,998	out-of-domain	numeric or normalized match

Answer Matching and Metrics. The predicted answer is the text inside the `<answer>` tag of the last action. Before exact or numeric matching, an answer longer than 120 characters is cut to

its first line. Two numbers match when their relative difference is at most 1%. Text answers match after lowercasing and removing articles and punctuation, and an answer with several items matches when its set of items equals that of the gold answer. FinQA stores a percentage either as a fraction or as a number of percent, so a FinQA answer also matches when 100 times or one hundredth of it matches the gold answer. TableBench follows its official rule by subtype. DescriptiveAnalysis, AnomalyDetection, and CausalAnalysis are scored by the ROUGE-L F1 between the answer and the gold answer, CorrelationAnalysis, TrendForecasting, and StatisticalAnalysis accept a relative error of 10% in each field, and the other subtypes use the rules above. The accuracy of a benchmark is the mean score over its questions, and Table 1 averages it over the training seeds of each method.

B.2 Rewards and Advantages

The training reward of Eq. (5) uses the matching rules of Appendix B.1, including the subtype rules of TableBench, with one change. For the three TableBench subtypes scored by ROUGE-L, the first two cases of Eq. (5) are replaced by the matching score, so a valid trajectory receives its ROUGE-L F1, a value in $[0, 1]$, as the reward instead of 1 or 0. The untrained agent almost never matches these sentence answers in full, so a binary reward would give their groups zero advantage. Invalid trajectories still receive -1 on every benchmark and subtype, below every valid trajectory.

The trajectory advantage of Eq. (1) divides by the standard deviation σ_R of the rewards in the group, to which we add 10^{-4} for numerical stability. When all trajectories of a group receive the same reward, $\sigma_R = 0$ and we set $A(\tau) = 0$ for every trajectory of the group, so the outcome gives this group no update direction. The relative term of Eq. (6) follows the same rule within $\mathcal{U}(t)$.

B.3 Policy Models and Multi-Turn Baselines

Policy Models. All methods start from the instruction checkpoints of Qwen2.5-Coder-3B-Instruct, Qwen2.5-Coder-7B-Instruct, and Qwen2.5-Coder-14B-Instruct (Hui et al., 2024) and train all parameters. The multi-turn methods share the prompt of Appendix B.8, the environment of Appendix B.7, the training pool, and the hyperparameters of Table 5 unless noted below.

Base. The untrained model acts as an agent with the same prompt, environment, and generation limits as the trained agents, which gives the accuracy of the prompt and the environment alone.

GRPO. GRPO (Shao et al., 2024) optimizes Eq. (11) with the trajectory advantage $A(\tau)$ for every action token, so all turns and tokens of a trajectory share one scalar credit.

TaREx. TaREx (Lei et al., 2026) releases no code, so we implement its recipe in our environment. The reward is $0.1R_{\text{fmt}}+0.9R_{\text{ans}}$, where $R_{\text{fmt}} = 1$ for a valid trajectory with a non-empty answer and $R_{\text{ans}} = 1$ when this answer is also correct. The objective uses asymmetric clipping with $\epsilon_{\text{low}} = 0.2$ and $\epsilon_{\text{high}} = 0.28$, averages the loss over tokens, and has no KL term. Three filters remove a trajectory that fills the context window without an answer, a trajectory with a turn that neither runs code nor answers, and a group whose remaining trajectories all receive the same reward.

GiGPO. GiGPO (Feng et al., 2025b) is the turn-level credit of Eq. (6) alone, with $\gamma = 0.95$ and $\omega = 1$. It equals TACO without the intra-turn credit, which is the second row of Table 2.

MT-GRPO. MT-GRPO (Wei et al., 2025) adds a turn-level reward to the outcome reward. A code turn receives a turn reward of 1 when its code runs without error and 0 otherwise. The turn rewards are normalized across the trajectories of a group at the same turn index, and the result is added to $A(\tau)$ for the tokens of that turn only, with equal weight as in the original paper.

GTPO. GTPO (Tan et al., 2025) shapes the reward of each token with the entropy of the policy at that position. A token of a correct trajectory receives an extra reward proportional to its entropy relative to the other correct trajectories of the group at the same position, and a token of an incorrect trajectory receives an extra penalty proportional to the inverse of its entropy. We use the coefficients $a_1 = 1$ and $a_2 = 0.1$ of the original paper and skip observation tokens when counting positions.

KTAE. KTAE (Sun et al., 2025) counts, within each group, how often each token occurs in correct and in incorrect trajectories and turns these counts into a key-token score with statistical tests of

association. The credit of a token is the trajectory advantage plus its key-token score. We use the released implementation with its default arguments over the action tokens of each group.

B.4 Combinations of GiGPO and Token-Level Methods

The two combinations in Figure 5 keep the turn-level credit A_t of GiGPO and replace the hindsight weight $w_{t,i}$ of Eq. (8) with a token-level credit. Within each turn, we map the GTPO or KTAE credit of the action tokens linearly onto $[1-\varepsilon_w, 1+\varepsilon_w]$, so that it has the same range as $w_{t,i}$. The token mask, the schedule of α, and all other settings follow TACO, so the two combinations differ from TACO only in the source of the intra-turn signal, which is the probability gap for TACO.

B.5 Single-Turn Baselines

The single-turn methods in Figure 6 see the whole table in the prompt, reason, and answer in one response without running code. They use the prompt of Appendix B.8, at most 1,024 tokens per response in training and 2,048 at evaluation, the same training questions as TACO, and the same checkpoint selection. Base is the untrained 7B model with this prompt. SFT follows Yang et al. (2025) and fine-tunes the model on reasoning traces for the training questions that end in the gold answer. RL follows the region-based recipe of Wu et al. (2025b), in which the model first names the rows and columns it will use and then answers. Its reward mixes the accuracy of the answer with the overlap between the named region and a reference region, and the weight of the region term decays as $0.3\exp(-9\times10^{-4}\,k)$ over training steps k, with a consistency penalty of weight 0.1.

B.6 Training and Hyperparameters

Each trained multi-turn agent runs for 1,000 optimizer steps with two questions per step and $N=8$ trajectories per question. A checkpoint is saved every 200 steps, and we select the checkpoint with the highest mean accuracy on two validation sets, 500 held-out TableInstruct questions and 495 held-out WikiTQ questions. Both validation sets are disjoint from the training pool and from every test set, and the test sets are used only for the final evaluation. Evaluation uses greedy decoding with the same prompt, environment, and generation limits as training. Table 5 lists the hyperparameters, which are the same at 3B, 7B, and 14B and for every multi-turn method unless Appendix B.3 states otherwise, as for the reward, the clipping, and the KL term of the TaREx recipe.

B.7 Execution Environment

Each code block runs in a fresh Python process in which the table is loaded as the DataFrame `df`, with pandas and NumPy imported as `pd` and `np`. Variables do not persist across turns, so later turns see earlier results only through the observations. A code block that runs longer than 10 seconds or uses more than 2 GB of memory is stopped and returns an error. The observation contains the printed output of the code and, when the last expression has a value that is not printed, that value after a `[value]` tag. For a failed block, it contains the error type and message after an `[error]` tag and the failing line after an `[at]` tag. An observation longer than 900 characters is cut to its first 900 characters and ends with a short note that gives its original length.

B.8 Prompts and Action Format

System Prompt. Every multi-turn method uses the system prompt below. It defines the three tags of an action, the observation returned by the environment, and the form of the final answer.

Table 5: Hyperparameters. The rollout, optimization, and evaluation settings are shared by all multi-turn methods at every model scale, and the last group applies to TACO only.

Rollout	
Group size N	8
Questions per step	2
Sampling temperature and top-p	1.0 and 0.95
Turn limit	none
Max tokens per action	4,096
Max sequence length	16,384
Table characters in the prompt	8,000, then the first and last five rows
Context characters in the prompt	2,200
Observation characters	900
Code time and memory limits	10 s and 2 GB
Optimization	
Optimizer	AdamW
Learning rate	1×10^{-6} with 5 warmup steps
Gradient clipping	1.0
Optimizer steps	1,000
Clip range ϵ	0.2
KL coefficient β	0.04
Checkpoint interval	200 steps
Evaluation	
Decoding	greedy
Credit assignment	
Discount γ	0.95
Relative weight ω	1
Teacher synchronization period P	10 steps
Weight bound ε_w	0.5
Temperature κ	1
Mixing coefficient α	$1 \to 0$ linearly over the first 60% of steps

System prompt of the multi-turn agent

```
You are a data analyst. You answer questions about a table by writing and running
Python.

You work in turns. In each turn you either run code or give the final answer:

<think> your reasoning </think>
<code>
# Python. The table shown below IS already loaded as a pandas DataFrame named `df`,
# with exactly those columns and rows. pandas is `pd`, numpy is `np`.
# print() what you want.
</code>

After a <code> block the system runs it and replies with an <observation> containing
its output or its error. Use that to check your work or fix your code, then take
another turn.

When you are sure, end with the final answer instead of code:

<think> your reasoning </think>
<answer> the answer </answer>

Rules:
- Exactly one <code> block or one <answer> per turn, always preceded by <think>.
- Never write an <observation> yourself; only the system does.
- Base the answer on values you actually computed and saw in an observation.
- The <answer> must be the bare answer only (a number, a word, or a short list) with
  no explanation, no units unless the question asks for them, and no surrounding
  sentence.
- If the answer has several parts, give them in the order the question asks, separated
  by ", ".
- Write numbers as plain numerals: no thousands separators, no currency symbol, and no
  unit unless the question asks for one.
- If the question asks for a category AND a value, give both, category first:
  `<category>, <value>`.
```

A TableBench question also receives the official answer format of its subtype, which TableBench provides for every subtype and which we rewrite only to use the <answer> tag. For the three subtypes whose gold answers are sentences, this format asks for a sentence and overrides the rule of a bare answer. The box below shows the format of CorrelationAnalysis as an example.

Answer format appended for a TableBench question of subtype CorrelationAnalysis

```
Answer format for this question type (TableBench's official rule for it; where it asks
for a sentence, it overrides the bare-answer rule above):
[Answer Format]
<answer>CorrelationRelation, CorrelationCoefficient</answer>

[Answer Examples]
<answer>No correlation, 0.22</answer>
<answer>Strong positive correlation, 0.82</answer>

The <answer> can only be in the "CorrelationRelation, CorrelationCoefficient." form, no
other form.
Ensure that: the correlation coefficient should be a float number with two decimal
places; the correlation relation can only be "No correlation" with the correlation
coefficient between -0.3 to +0.3, "Weak positive correlation" with the correlation
coefficient between +0.3 to +0.7, "Weak negative correlation" with the correlation
coefficient between -0.3 to -0.7, "Strong positive correlation" with the correlation
coefficient between +0.7 to +1, or "Strong negative correlation" with the correlation
coefficient between -0.7 to -1.
```

User Prompt and Table Rendering. The user prompt contains the context, the table, and the question. The table is rendered in Markdown after its shape and column names. When the rendered table exceeds 8,000 characters, the prompt shows only its first and last five rows together with the number of rows left out, while `df` still holds the whole table. The context is the report text of a FinQA question or the caption of a HiTab table, cut to 2,200 characters, and it is empty for the questions of TableBench and WikiTQ, which provide only the table and the question.

User prompt of the multi-turn agent

```
Context:
{report text or table caption, if any}

Table:
shape: {R} rows x {C} cols
columns: [{column names}]
table:
{the table in Markdown}

Question: {question}

Work it out with Python over `df`, then give the final answer.
```

Single-Turn Prompt. The single-turn methods of Figure 6 have no code tool. They use the system prompt below and the user prompt above. The last line of the user prompt, which asks the model to use Python, is replaced by "Reason it through step by step from the table, then give the final answer."

System prompt of the single-turn methods

```
You are a data analyst. You answer questions about a table by reasoning over the table
shown below. You have no code tool: read the table, reason step by step, and answer in
one reply.

<think> your reasoning </think>
<answer> the answer </answer>

Rules:
- Exactly one <think> block followed by exactly one <answer> block; nothing after
  </answer>.
- Base the answer on values that are actually in the table.
- The <answer> must be the bare answer only (a number, a word, or a short list) with
  no explanation, no units unless the question asks for them, and no surrounding
  sentence.
```

Parsing and Validity. The whole trajectory is one assistant message, in which each observation follows the code block that produced it. An action is parsed as a code action when it contains a closed <code> block and as an answer action when it contains an <answer> tag. An action with neither, such as an unclosed code block, is a malformed action and makes the trajectory invalid. A

trajectory is valid when it has no malformed action, ends with an answer that can be extracted, and contains at least one code block that runs without error. The loss of Eq. (11) covers only the action tokens, since the observation tokens come from the execution environment.

B.9 HINDSIGHT CONTEXT

Teacher Input. The teacher scores the same action tokens as the policy, in the same order, so the two log-probabilities align token by token in Eq. (7). Its input is the policy input with one hindsight block inserted right before each action, and the observations stay where they are. Every block starts with a marker line that the policy never sees. For a code action, the block states whether the code runs and then gives its printed output and value, or the error message and the failing line. For the answer action, the block gives the verdict of the answer checker and the gold answer. The printed output is cut to 400 characters and the value to 200. When the context includes a correct trajectory, its block comes first, in front of the first action. The box below shows the teacher input for an incorrect trajectory on a TableBench question about the average height of five buildings, with the reasoning and the code shortened and the system and user prompts omitted.

Teacher input for an incorrect trajectory on TableBench

```
{system prompt} {user prompt}
[TRAINING-ONLY EXECUTION HINDSIGHT -- a sibling attempt at this question that was judged
CORRECT; not visible to the model at inference]
<code>
top_5_heights = df['height'].head(5).str.replace(
    ' m \([^)]+\)', '', regex=True).astype(float)
...
</code>
<answer>92.2</answer>
[TRAINING-ONLY EXECUTION HINDSIGHT -- not visible to the model at inference]
this action's code will execute with status: OK
it will print: 87491.2
<think> ... </think><code> top_5_buildings = df.head(5) ... </code>   <- turn 1, scored
<observation>
87491.2
</observation>
[TRAINING-ONLY EXECUTION HINDSIGHT -- not visible to the model at inference]
outcome of the whole trajectory: INCORRECT
reference answer: 92.2
<answer>87.4912</answer>                                              <- turn 2, scored
```

Correct Trajectory. The correct trajectory is taken from the group of the scored trajectory. Among the correct trajectories of the group other than the scored one, we take the one with the lowest sample index, so the choice is deterministic. We keep its code blocks and final answer, remove its reasoning and observations, and cut it to 1,500 characters. When the group has no other correct trajectory, the context contains only the execution feedback of the turn, which every sampled trajectory has. In the 7B training runs, 86.1% of the groups contain at least one correct trajectory, so every incorrect trajectory of these groups receives one in its hindsight context.

Separation from the Policy. The marker in every hindsight block serves as a check. When the prompts of the policy are built, the training code asserts that the marker does not occur in any of them. The teacher shares the tokenizer of the policy, and its parameters are copied from the policy every $P = 10$ optimizer steps. The teacher runs without gradients, so hindsight changes only the weights $w_{t,i}$ of the update and never enters the input of the policy, in training or at test time.

Variants of the Hindsight Context. Table 3 compares six contexts, which are built from the lines of the teacher input above. Output only keeps the printed output and value, or the error message and line, of a code turn, and only the gold answer for the answer turn. Immediate feedback adds the status line of a code turn and the verdict of the answer turn. Full feedback places the immediate feedback of every turn of the trajectory in front of each action, so the teacher also reads the results of later turns. Correct traj. only gives the correct trajectory without any feedback, and the last two variants add it to the full or the immediate feedback. All variants keep the rest of TACO fixed.

C Additional Details of the Pilot Study

C.1 Settings

The pilot study of Section 3 uses the runs of Table 1, so GRPO and the four credit assignment methods share the training pool, reward, prompt, and all settings of Appendix B. Figure 1 shows their TableBench and FinQA accuracy, and the rest of this section details Figures 2 and 3.

C.2 Error Annotation

Figure 2 (Left) uses questions from TableBench and FinQA that all five methods of Figure 1 answer incorrectly, from which we sample 500 incorrect trajectories. For each trajectory, the annotators read the question, the table, every action and observation, and the gold answer, and mark the operation that first makes the answer wrong. The error is a turn-level error when the marked turn contains no correct operation or should not be taken at all, and an intra-turn error when the marked turn also contains correct operations. Other errors are those with no wrong table operation, such as a correct value that the answer copies incorrectly or writes in the wrong format.

C.3 Operation Counting

Figure 2 (Right) counts the table operations in each code turn of the incorrect trajectories. We skip comment lines and print statements and count three kinds of operations in the remaining lines. A row filter is an index expression that contains a comparison or a call such as `isin`, `between`, or `str.contains`. A column selection is an index expression that contains a quoted column name. A table method is a call such as `groupby`, `sort_values`, `merge`, `sum`, `mean`, or `idxmax`, from a fixed list of 54 pandas methods that select, reshape, clean, or aggregate data.

C.4 Token Credit of GTPO and KTAE

Figure 3 (Left) compares the credit that GTPO and KTAE give to the tokens of correct and of incorrect operations. We use training trajectories of the 3B GRPO agent from its last 200 optimizer steps, with eight trajectories per question. In an incorrect trajectory, an operation of a code turn, extracted as in Appendix C.3, is labeled correct when the same operation occurs in a correct trajectory of the same question from another 3B run, and incorrect otherwise. Labels from other runs keep KTAE from scoring the same co-occurrence that defines the label. We keep the turns that contain operations of both kinds, and each token takes the label of the operation it belongs to. We compute the token credit of GTPO and KTAE as in Appendix B.3, with the entropy of GTPO taken from the saved checkpoint closest to the step that sampled each trajectory. We then normalize the credit of the action tokens of each turn to $[0, 1]$ by its minimum and maximum, and average it over the tokens of each label. The figure reports the mean over the 457 kept turns with its standard error.

C.5 Hindsight Gap

Figure 3 (Right) compares the gap of Eq. (7) between the tokens of correct and of incorrect operations within the same turns. We use the 457 turns of Appendix C.4, in which every token takes the label of its operation. A hindsight teacher copied from the 3B checkpoint closest to the step that sampled each turn scores the turn with its immediate feedback as the context. For each turn, we average the gap over the code tokens of each label, so every turn gives one value for its correct operations and one for its incorrect operations. Values beyond ± 0.12 go to the outermost bins, and 10.8% of the incorrect and 1.4% of the correct values lie below -0.12.

D Additional Experimental Details

Table 6 lists how each variant of Table 2 changes the equations of Section 4. Each variant changes one component and keeps every other setting of TACO, including the hindsight context, the training pool, the training seeds, and the checkpoint selection on the validation sets.

Table 6: Variants of the ablation study in Table 2. Each row gives the change to the equations of Section 4, and all other settings, including the hindsight context, follow TACO.

Variant	Change
w/o turn-level credit	A_t is replaced by $A(\tau)$ in Eqs. (8) and (10)
w/o intra-turn credit	$\alpha = 0$ at every step, which gives GiGPO
w/o token mask	$m_{t,i} = 1$ for every action token, including the reasoning z_t
w/o weight clip	$w_{t,i} = \exp\big(\mathrm{sg}[\mathrm{sgn}(A_t)\Delta_{t,i}/\kappa]\big)$ without the bound ε_w
w/o α decay	$\alpha = 1$ at every step