AGENTREROUTE: TRAINING LLMS TO CORRECT AGENT TRAJECTORIES WITH REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Once a large language model agent makes an erroneous decision early in a task, the resulting error can accumulate along its autoregressive trajectory and ultimately lead to task failure. Dominant self-correction methods rely on reflection and feedback to improve outputs, but the model's self-evaluation is often influenced by its original erroneous judgments, making it difficult to correct accumulated errors. To this end, external assistance methods introduce dedicated modules for diagnosis and feedback to correct accumulated errors. However, these methods often rely on heuristic rules or prompt-based judgments, making it difficult to achieve both accurate rollback and effective correction. To address this issue, we propose AGENTREROUTE, an externally assisted reinforcement learning framework that uses task outcomes as reward signals to **jointly optimize rollback control and corrective feedback**. Specifically, AGENTREROUTE freezes the action generator, Generator, and trains only an independent recovery controller, Rerouter. Given the task instruction, the active trajectory, and the recovery record, Rerouter decides whether to continue execution or select a rollback point and generate corrective feedback, thereby guiding Generator to regenerate subsequent actions. Extensive experiments on four question-answering tasks, WebShop, and the Game24 interactive environment show that AGENTREROUTE significantly improves task success rates while keeping Generator frozen, and that it can be combined with Reflexion for further gains. Further analysis shows that the learned recovery policy reduces repetitive behavior after rollback and maintains stable gains when reused across question-answering datasets and across Generators, demonstrating strong generalization and cross-model transferability. Our code and dataset are anonymously available at `https://anonymous.4open.science/r/AgentReroute-DCCA/`.

1 INTRODUCTION

In multi-step tasks such as retrieval-based question answering, mathematical reasoning, and web interaction, an early decision error by a large language model agent can propagate along its autoregressive execution trajectory, making subsequent recovery increasingly difficult and ultimately causing task failure (Li et al., 2025b; Zhang et al., 2026b). Self-correction methods improve outputs through reflection and feedback (Madaan et al., 2023; Shinn et al., 2023; Gou et al., 2024), but self-evaluation may inherit the model's original erroneous judgments and thus struggle to correct critical errors reliably without dependable external signals (Huang et al., 2024).

To address this limitation, externally assisted methods introduce independent modules for diagnosis and feedback (Yao et al., 2024; Xie et al., 2025), while trajectory rollback allows agents to resume task solving from earlier positions (Li et al., 2025b; Zhang et al., 2026b; Zhuang et al., 2026). However, these methods largely rely on prompting, rules, or predefined recovery mechanisms, making it difficult to jointly determine where to roll back and how to guide subsequent correction. As shown in Figure 1, rolling back too far can discard useful progress, whereas rolling back too little can retain the source of failure. Even when the rollback point is appropriate, insufficiently targeted feedback may cause the same error to recur.

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.

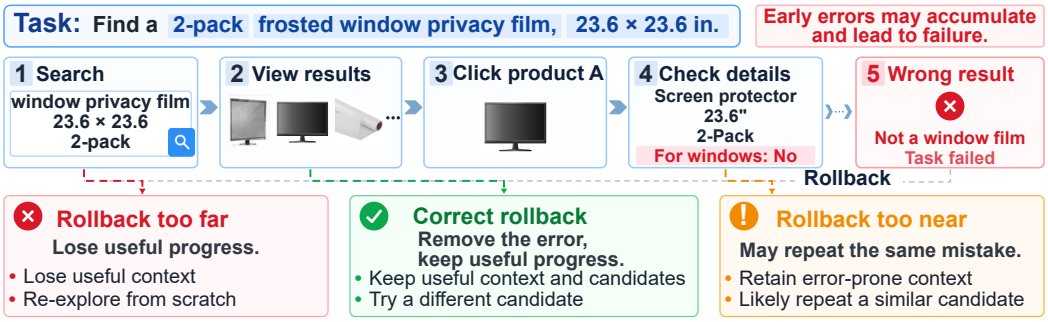

Figure 1: Illustration of different rollback choices. Rolling back too far discards useful progress, while rolling back too near retains error-prone context and may repeat the same mistake. An appropriate rollback preserves useful progress while removing the error.

We therefore propose AgentReroute, an externally assisted reinforcement learning framework that uses task outcomes as reward signals to optimize rollback control and corrective feedback jointly. AgentReroute **keeps the action-generating Generator frozen and trains only an independent Rerouter**. Given the task instruction, the current active trajectory, and the recovery record, Rerouter produces a structured recovery decision: it either selects CONTINUE to retain the current branch and proceed with execution, or selects ROLLBACK-TO-k to return to a previous action position and generate corrective feedback that guides Generator to regenerate from that point. During training, we sample multiple complete executions for the same task and use their final task outcomes to jointly optimize recovery control and feedback, without requiring step-level recovery labels or reference feedback text.

We evaluate AgentReroute on four question-answering datasets, the WebShop web shopping environment, and the Game24 interactive arithmetic environment. Experiments show that the learned recovery policy substantially improves task performance while Generator remains frozen, and can be combined with Reflexion for further gains. Behavioral analysis further shows that the trained Rerouter reduces repetition of previous behavior after rollback. Moreover, the same recovery policy maintains stable improvements when reused across question-answering datasets and when paired with different Generators. These results indicate that an independently learned recovery policy can effectively steer existing LLM agents away from failing trajectories and provide reusable recovery behavior across tasks and models.

2 Related Work

2.1 Error Correction in Language Models and Agents

Existing error correction methods broadly fall into self-correction and externally assisted correction. Self-correction methods revise outputs through iterative reflection, tool-based verification, and execution feedback (Madaan et al., 2023; Shinn et al., 2023; Gou et al., 2024; Chen et al., 2024). Iterative training and reinforcement learning further strengthen reflection, revision, and backtracking capabilities (Qu et al., 2024; Yuan et al., 2025; Kumar et al., 2025; Zhang et al., 2026c). Without reliable external feedback, however, self-evaluation may rely on the same erroneous judgments and struggle to correct critical errors reliably (Huang et al., 2024).

Externally assisted correction uses independent reflection or critic models to provide feedback (Yao et al., 2024; Xie et al., 2025), or combines error detection, rollback, and action rewriting to revise execution trajectories (Li et al., 2025b; Zhang et al., 2026b; Wu et al., 2025). Checkpoints, semantic recoverability constraints, and sandbox state restoration provide further support for rollback (Zhuang et al., 2026; Yang et al., 2026; Dong et al., 2026). These approaches largely rely on prompting or rules for recovery control, or focus learning on feedback content. Such strategies can retain errors or discard useful progress. AgentReroute instead keeps Generator frozen and trains an independent Rerouter to jointly learn rollback control and corrective feedback from task outcomes, guiding Generator to resume execution.

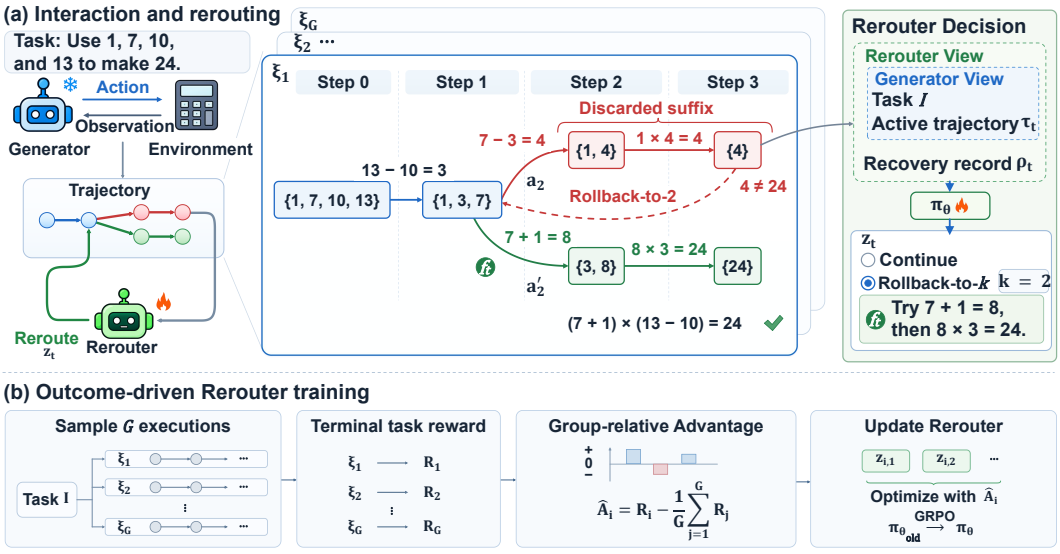

Figure 2: Overview of AgentReroute. (a) A frozen Generator interacts with the environment. Rerouter uses the task instruction, active trajectory, and recovery record to choose whether to continue or roll back with corrective feedback. In the Game24 example, rollback to $k = 2$ retains the first action and guides regeneration. (b) Terminal rewards from G complete executions of the same task yield group-mean-centered advantages. All Rerouter responses within an execution share its advantage for GRPO updates.

2.2 Policy Learning from Outcome Feedback

Outcome feedback links intermediate decisions to final task performance. DeepSeek-R1 trains reasoning with verifiable rewards (DeepSeek-AI, 2025). Search-R1, ReSearch, and R1-Searcher use answer outcomes to learn search policies (Jin et al., 2025; Chen et al., 2025; Song et al., 2025), while ReTool optimizes tool use based on answer correctness (Feng et al., 2025). For auxiliary decisions, Memory-R1 learns memory operations from question-answering outcomes (Yan et al., 2025), and Router-R1 combines correctness, format, and cost signals to learn multi-round model routing and answer aggregation (Zhang et al., 2025). Although these methods improve task decisions through outcome feedback, they primarily advance execution along the current trajectory, leaving earlier erroneous decisions difficult to remove from the execution history. AgentReroute instead uses outcome feedback to learn an external recovery policy that rolls back erroneous branches and guides subsequent execution.

3 AgentReroute

AgentReroute formulates trajectory correction in multi-step interactive tasks as learning an independent recovery policy. Figure 2 summarizes the interaction process and outcome-driven Rerouter training. To coordinate rollback control and corrective feedback, a dedicated Rerouter generates structured recovery decisions (Section 3.2), which guide subsequent Generator actions through trajectory recovery and feedback (Section 3.3). Task outcomes provide an overall evaluation of the recovery policy's contribution to task completion and guide Rerouter training through group relative policy optimization (Section 3.4).

3.1 Problem Formulation

We consider multi-step interactive tasks whose goals are specified by a natural language instruction I. An agent seeks an outcome that satisfies the task requirements through a sequence of action–observation interactions (Li et al., 2025b; Zhuang et al., 2026). The environment $\mathcal{E}$ starts in state s_0 and provides an initial observation o_0. The agent generates actions based on the instruction and prior interactions. After executing action a_j in state s_{j-1}, the environment transitions to s_j and returns

observation o_j:

$$(s_j, o_j) \sim \mathcal{T}_{\mathcal{E}}(\cdot \mid s_{j-1}, a_j), \tag{1}$$

where $\mathcal{T}_{\mathcal{E}}$ specifies the environment's state transition and observation generation mechanism, and j indexes action positions within the current branch.

At recovery decision t, let n_t denote the number of actions in the current branch. The active trajectory alternates between actions and environment observations:

$$\tau_t = (o_0, a_1, o_1, \ldots, a_{n_t}, o_{n_t}), \tag{2}$$

where τ_t retains the execution branch used for subsequent task solving. Here, t indexes recovery decisions rather than action positions, and the branch length n_t can decrease after rollback.

We study trajectory correction in this interaction process and assume that the environment state corresponding to a retained prefix can be reconstructed at recoverable positions. Let ξ denote a complete execution from task initialization to termination, with final task performance evaluated by the scoring function $S(I, \xi)$. A complete execution includes interactions and recovery decisions before and after rollback, not just the active trajectory retained at termination. Our goal is to learn a policy that improves task outcomes through rollback control and corrective feedback while keeping the action generator fixed.

3.2 Structured Recovery Decisions

The rollback point determines the context retained for regeneration, while corrective feedback guides subsequent actions within that context. AgentReroute therefore represents control decisions and corrective feedback jointly as structured recovery decisions generated by Rerouter π_θ. Generator g_ϕ generates actions and interacts with the environment $\mathcal{E}$, while Rerouter makes recovery decisions after the environment returns observations. Training updates only θ and keeps ϕ frozen.

At decision t, Rerouter conditions on the task instruction I, the active trajectory τ_t, and the recovery record ρ_t. The active trajectory is the current execution branch defined in Equation (2). The recovery record accumulates removed trajectory suffixes and previous structured recovery decisions as evidence for subsequent decisions.

Given this information, Rerouter jointly generates a control decision d_t and optional natural language corrective feedback f_t, forming a structured recovery decision z_t:

$$z_t = (d_t, f_t) \sim \pi_\theta(\cdot \mid I, \tau_t, \rho_t). \tag{3}$$

The control decision is chosen from the following set:

$$d_t \in \{\text{Continue}\} \cup \{\text{Rollback-to-}k \mid k \in \mathcal{K}_t\}, \tag{4}$$

where $\mathcal{K}_t \subseteq \{1, \ldots, n_t\}$ is the set of recoverable action positions in the current trajectory. Continue retains the active trajectory and lets Generator produce the next action. Rollback-to-k rolls back execution, with k specifying the first action position to regenerate. The feedback f_t guides regeneration after rollback.

3.3 Feedback-Guided Trajectory Recovery

As illustrated in Figure 2(a), the control decision determines how the active trajectory is updated, and corrective feedback conditions regeneration after rollback. When $d_t = \text{Continue}$, the Generator produces the next action given the task instruction I and the active trajectory τ_t. The environment executes this action and returns an observation according to Equation (1). Appending the new action–observation pair yields the updated active trajectory:

$$\begin{aligned} a_{n_t+1} &\sim g_\phi(\cdot \mid I, \tau_t), \\ \tau_{t+1} &= \tau_t \oplus (a_{n_t+1}, o_{n_t+1}), \end{aligned} \tag{5}$$

where $\oplus$ denotes sequence concatenation.

When $d_t = \text{Rollback-to-}k$, AgentReroute follows GA-Rollback in preserving the trajectory prefix and regenerating its suffix (Li et al., 2025b). It retains the prefix $\tau_t^{<k}$ consisting of the initial observation and the first $k-1$ action–observation pairs, removes the old suffix starting at a_k, and restores the

environment to the corresponding state s_{k-1}. Generator then receives f_t and regenerates from action position k:

$$\begin{aligned} a'_k &\sim g_\phi(\cdot \mid I, \tau_t^{<k}, f_t), \\ \tau_{t+1} &= \tau_t^{<k} \oplus (a'_k, o'_k), \end{aligned} \tag{6}$$

where o'_k is the observation returned after the environment executes a'_k from s_{k-1} according to Equation (1). The feedback f_t guides the first action generated after recovery; subsequent actions are generated from the updated active trajectory.

After each decision, AgentReroute forms ρ_{t+1} by appending z_t to ρ_t and, if rollback occurs, also storing the removed suffix. Rerouter then conditions its next decision on I, τ_{t+1}, and ρ_{t+1}. This interaction continues until task termination, forming a complete execution ξ. Appendix A shows the training template and its decision format.

3.4 Outcome-Driven Recovery Policy Optimization

Rollback and corrective feedback alter subsequent execution, so we evaluate the recovery policy by its overall contribution to task completion. As shown in Figure 2(b), AgentReroute uses final task outcomes from complete executions as rewards to optimize control decisions and corrective feedback jointly. For each task instruction I, we keep Generator frozen and sample G complete executions $\{\xi_i\}_{i=1}^{G}$ under the old recovery policy $\pi_{\theta_{\text{old}}}$ using the interaction procedure in Section 3.3. Each execution receives a scalar return based on its final task outcome: $R_i = S(I, \xi_i)$, where S is the final task scoring function defined in Section 3.1.

We use group relative policy optimization (GRPO) (Shao et al., 2024) to turn task returns into learning signals for the recovery policy. Using the mean return within each group as the baseline, we compute the relative advantage for execution i as $\widehat{A}_i = R_i - \frac{1}{G}\sum_{j=1}^{G} R_j$. We center returns by the group mean without dividing by the within-group standard deviation. All Rerouter responses within execution i share $\widehat{A}_i$, so both control decisions and corrective feedback receive a learning signal from the same final task outcome.

Let N_i be the total number of tokens across all Rerouter responses in execution i. Denote the ℓ-th response token by $y_{i\ell}$ and its actual generation context by $h_{i\ell}$. The probability ratio between the new and old policies for this token is

$$r_{i\ell}(\theta) = \frac{\pi_\theta(y_{i\ell} \mid h_{i\ell})}{\pi_{\theta_{\text{old}}}(y_{i\ell} \mid h_{i\ell})}. \tag{7}$$

Using the execution-level advantage, the clipped policy term for Rerouter is

$$J_{\text{clip}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{N_i}\sum_{\ell=1}^{N_i}\min\Big\{r_{i\ell}(\theta)\widehat{A}_i, \text{clip}\big(r_{i\ell}(\theta), 1-\epsilon, 1+\epsilon\big)\widehat{A}_i\Big\}\right], \tag{8}$$

where ϵ is the clipping threshold, and the expectation is over training tasks and complete executions sampled under the old recovery policy. This policy term first averages over all Rerouter response tokens within each complete execution and then across executions for the same task. Importantly, optimization retains Rerouter responses from the entire execution, including those generated before a branch is discarded. Each response token is evaluated under its original generation context, not under the shortened trajectory produced by a later rollback. The shared terminal advantage therefore trains the sequence of recovery decisions collectively, rather than assigning a separate correctness label to each rollback point or feedback message. Optimization updates only the Rerouter parameters θ, while Generator remains frozen. At inference time, the learned Rerouter guides Generator through the same interaction procedure.

4 Experiments

We evaluate AgentReroute on WebShop and Game24 with two frozen Generators, and on four retrieval-based question-answering datasets using a single Rerouter trained on MuSiQue to examine cross-dataset generalization.

4.1 Experimental Setup

Tasks and evaluation metrics. WebShop (Yao et al., 2022) requires agents to complete shopping tasks through web interaction. We use 750 tasks for training and 500 for evaluation, reporting the mean terminal environment score (Score) and success rate (SR). The training reward is the raw terminal environment score. Game24 (Yao et al., 2023a) requires agents to reach 24 through arithmetic operations. We use 250 tasks for training and 500 for evaluation, report SR, and use a binary task-success indicator as the training reward.

We additionally evaluate retrieval-based question answering on HotpotQA (Yang et al., 2018), 2WikiMultihopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023). We train the Rerouter on 1,000 MuSiQue questions and evaluate the same checkpoint on all four datasets, using 500, 500, 500, and 125 questions, respectively. We report exact match (EM) and token-level F1 (F1) for final answers, using F1 as the training reward.

Models and training. We use Qwen3-8B and Llama-3.2-3B-Instruct as frozen Generators for WebShop and Game24, and Qwen3-8B for question answering. All Rerouters are initialized from Qwen3-4B-Instruct-2507, and we train a separate Rerouter for each task–Generator pair using GRPO (Shao et al., 2024). We set the Generator temperature to 0.1 across all three task families, while the Rerouter uses greedy decoding during validation and testing.

Baselines. For WebShop and Game24, we compare against task-specific agent baselines, including Act-only, ReAct (Yao et al., 2023b), Reflexion (Shinn et al., 2023), and GA-Rollback (Li et al., 2025b), together with ReAct + Reflexion, ReAct + GA-Rollback, and Reflexion + GA-Rollback. We additionally include AgentSquare (Shang et al., 2025) and ACE (Zhang et al., 2026a) as general agent optimization baselines, and evaluate AgentReroute + Reflexion to study its compatibility with between-attempt reflection.

For question answering, we compare against retrieval-free generation (Base model), single-round retrieval (Naive RAG), multi-step retrieval methods including Iter-RetGen (Shao et al., 2023), IR-CoT (Trivedi et al., 2023), and Search-o1 (Li et al., 2025a), and the general agent optimization methods AgentSquare and ACE.

4.2 Main Results

Overall performance. Table 1 shows that AgentReroute outperforms the baselines in task success rate on both WebShop and Game24 with either Generator. The consistent gains across model families and interaction environments support learning recovery policies from task outcomes rather than relying on evaluated rule-based or prompted recovery methods. The advantage is particularly pronounced on Game24, where each arithmetic operation changes the numbers available for subsequent steps, making recovery sensitive to the state from which execution resumes.

Combination with Reflexion. Combining AgentReroute with Reflexion further improves every reported metric under both Generators and achieves the best overall results. This supports the complementarity of the two recovery mechanisms: AgentReroute corrects the trajectory within an ongoing attempt, while Reflexion uses experience from one attempt to guide the next.

4.3 Cross-Dataset Generalization on Question Answering

As shown in Table 2, AgentReroute achieves the highest EM and F1 scores on all four question-answering benchmarks, including MuSiQue, on which the Rerouter is trained, and three datasets that are not used for training. This comparison indicates that the improvement does not simply result from adding retrieval steps or applying a generic agent optimization procedure, but is associated with correcting unproductive intermediate trajectories.

The results on the three unseen datasets further show that the same Rerouter can be applied without dataset-specific training. The gains are particularly pronounced on HotpotQA and Bamboogle, while the method maintains the best performance on 2WikiMultihopQA, covering datasets with different question and evidence distributions. This pattern supports the view that the learned policy captures recovery behaviors that transfer across question-answering tasks, such as revising an unproductive

Table 1: Results on WebShop and Game24 with two frozen Generators. All values are percentages. Bold and underlined entries mark the best and second-best values in each column, excluding component variants.

	Qwen3-8B			**Llama-3.2-3B-Instruct**		
	WebShop		Game24	WebShop		Game24
Method	Score↑	SR↑	SR↑	Score↑	SR↑	SR↑
Task-specific agent baselines						
Act-only	61.75	29.0	3.2	15.55	7.4	0.4
ReAct	50.31	23.6	14.2	7.92	4.2	0.8
Reflexion	<u>67.77</u>	33.8	11.6	21.83	11.4	0.4
ReAct + Reflexion	58.08	31.6	30.0	16.74	8.6	0.6
GA-Rollback	51.99	30.4	59.6	26.31	15.0	6.6
ReAct + GA-Rollback	31.04	13.6	54.6	23.58	13.6	2.2
Reflexion + GA-Rollback	64.52	37.4	73.2	30.57	17.2	13.4
General agent optimization baselines						
AgentSquare	52.93	25.6	17.4	16.22	7.4	2.2
ACE	44.19	25.8	16.2	20.97	9.6	3.0
Ours						
AgentReroute (4B)	63.59	<u>37.6</u>	<u>77.8</u>	<u>53.94</u>	<u>23.8</u>	<u>52.8</u>
+ Reflexion	**70.48**	**43.8**	**83.6**	**63.67**	**29.4**	**65.8**
Component variants of AgentReroute						
w/o Feedback	58.18	35.0	3.0	27.84	13.2	6.2
w/o RL Training	47.73	29.8	15.6	28.59	13.2	9.0

Table 2: Results on four multi-hop question-answering benchmarks. EM and F1 are reported as percentages. Bold and underlined values indicate the best and second-best results, respectively.

Method	**HotpotQA**		**2Wiki**		**MuSiQue**		**Bamboogle**	
	EM	F1	EM	F1	EM	F1	EM	F1
Task-specific QA baselines								
Base model	15.40	24.39	18.80	25.81	1.20	8.74	8.00	16.30
Naive RAG	26.20	36.94	26.20	32.28	4.20	10.44	13.60	24.63
Iter-RetGen	26.60	38.89	28.40	34.21	7.40	14.57	17.60	27.89
IRCoT	29.60	39.34	20.60	28.41	6.80	10.84	24.00	32.23
Search-o1	29.20	42.83	<u>31.00</u>	<u>39.82</u>	10.20	<u>19.52</u>	<u>35.20</u>	<u>51.85</u>
General agent optimization baselines								
AgentSquare	<u>33.40</u>	<u>44.41</u>	23.80	28.85	<u>12.20</u>	19.49	29.60	39.02
ACE	25.60	34.87	12.00	18.06	7.80	14.44	16.00	24.98
Ours								
AgentReroute (4B)	**44.60**	**56.52**	**36.20**	**43.88**	**17.00**	**25.04**	**48.80**	**58.63**

retrieval attempt and continuing from a corrected trajectory. Overall, the results provide evidence that trajectory correction can serve as a transferable recovery capability for retrieval-based agents. Appendix D presents one representative recovery trajectory from each task family.

4.4 Ablation and Recovery Behavior Analysis

Effects of training and feedback. We separately remove reinforcement learning from Rerouter training and corrective feedback after rollback to examine their contributions. As shown in Table 1, either removal reduces performance across tasks and Generators. The untrained Rerouter uses the same recovery interface but falls short of the full method, supporting terminal task outcomes as

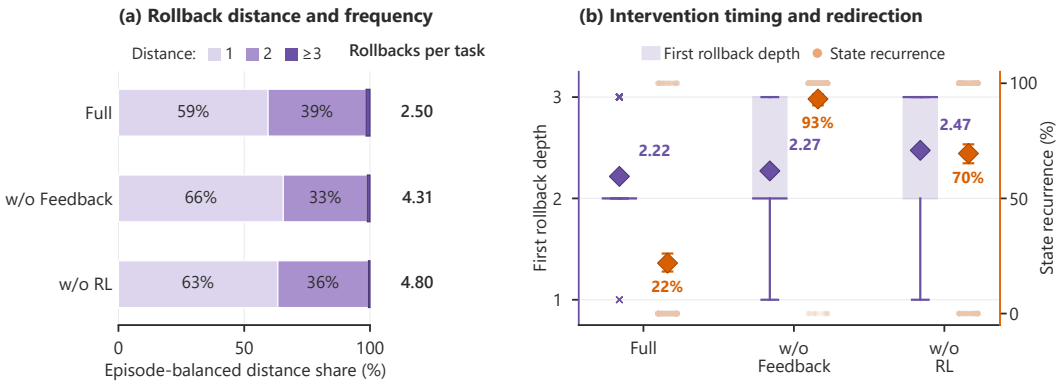

Figure 3: Recovery behavior on Game24 tasks. (a) Distribution of executed rollback distances and mean rollbacks per task. (b) Active-branch depth at the first rollback and recurrence of the discarded post-action state after replacement; diamonds denote means or rates, and orange intervals are 95% Wilson confidence intervals.

an effective training signal for recovery. Removing feedback causes a particularly large drop on Game24, indicating that returning to an earlier position and regenerating alone is insufficient for reliable correction.

Intervention frequency and rollback depth. To understand these differences, we compare the recovery behavior of the full method with both ablations. Figure 3(a) shows that the full method executes fewer rollbacks per task, while the rollback-distance distributions remain similar. The depth of the first rollback also differs only modestly in Figure 3(b). Together with the success rates, these observations show that better recovery can accompany fewer rollbacks, and that more frequent intervention does not necessarily improve task performance.

Post-rollback redirection. A clearer difference appears after regeneration. Without corrective feedback, the Generator frequently returns to the state produced by the discarded action; recurrence is also common without reinforcement learning. The full method substantially reduces this recurrence. Together with the ablation results, this supports the role of feedback in redirecting subsequent execution and suggests that recovery policy training helps reduce ineffective repeated attempts. State recurrence compares the state after the first valid replacement with that produced by the discarded action, and is computed only when both states are parseable. It measures local redirection, rather than terminal success; the three conditions are compared descriptively using independently sampled trajectories (Appendix C.1).

4.5 Cross-Generator Transfer and Training Dynamics

Cross-Generator transfer. We evaluate Rerouters trained with Qwen3-8B and Llama-3.2-3B-Instruct under both Generators to examine their dependence on the training-time Generator. As shown in Figure 4(a), both trained Rerouters outperform the untrained base Rerouter after switching Generators, demonstrating transfer between the evaluated models. Transfer is asymmetric: the Rerouter trained with Qwen performs comparably to the Rerouter trained with Llama when both are paired with Llama, whereas transfer in the reverse direction falls short of training directly with Qwen. Thus, among the two evaluated settings, training with Qwen yields a recovery policy that transfers more effectively across Generators.

Training dynamics. Figure 4(b) compares validation performance when the same Rerouter base model is trained with each frozen Generator. Both runs improve over initialization on the shared validation set, with more sustained gains under Qwen and larger fluctuations under Llama. These observations suggest that the Generator's trajectories influence recovery policy learning even when the Rerouter architecture and training objective remain fixed.

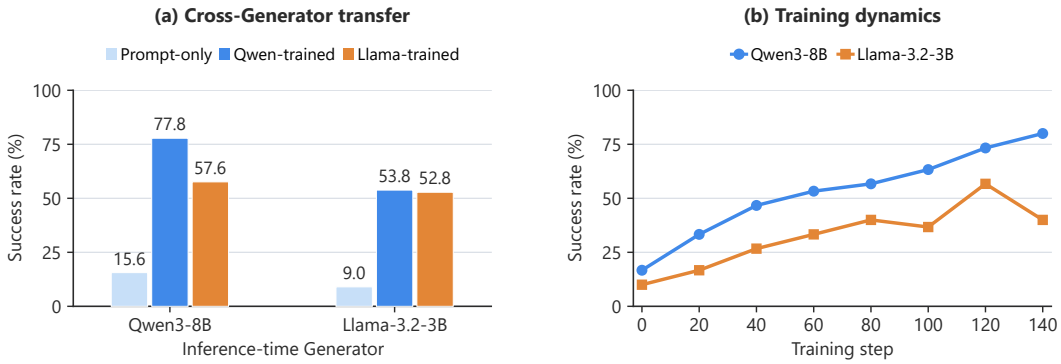

Figure 4: Cross-Generator transfer and training dynamics on Game24. (a) Test success rates for the prompt-only Rerouter and Rerouters trained with each frozen Generator. (b) Validation success rates during training with Qwen3-8B or Llama-3.2-3B as the frozen Generator. Both curves use the same validation set.

Table 3: Success rate and inference cost on Game24 with a fixed Llama-3.2-3B-Instruct Generator. Gen. and Aux. denote the average token consumption of the Generator and auxiliary models per task, respectively. Time denotes the mean end-to-end latency per task.

		Tokens / task (k)↓		
Method	SR (%)↑	Gen.↓	Aux.↓	Time (s)↓
Act-only	0.4	10.21	0	4.09
Reflexion	0.4	24.07	3.78	15.02
GA-Rollback	6.6	20.05	14.61	28.94
AgentReroute	52.8	7.57	19.18	74.58

4.6 Inference Cost

Table 3 compares task performance and inference cost with Llama fixed as the Generator. AgentReroute achieves higher success while consuming fewer Generator tokens. Its combined Generator and auxiliary token consumption is also lower than that of GA-Rollback. However, AgentReroute uses more auxiliary tokens and has the longest end-to-end latency among the methods compared. These results reveal a trade-off between task performance and response time: reduced Generator usage adds computation for recovery decisions and feedback, so lower token consumption does not translate into faster task completion.

5 Conclusion

We present AgentReroute, an externally assisted reinforcement learning framework that uses task outcomes as rewards to optimize targeted rollback control and corrective feedback jointly. The framework keeps the Generator frozen and trains an independent Rerouter to decide whether and where to roll back and to provide corrective feedback. Experiments on WebShop, Game24, and multi-hop question answering show that AgentReroute improves task performance and complements Reflexion. The recovery policy trained only on MuSiQue generalizes to unseen question-answering datasets, while cross-Generator experiments demonstrate transfer between the two evaluated Generators. Ablation and behavioral analyses further support the contributions of policy training and corrective feedback, indicating that they help reduce state recurrence after rollback and ineffective repeated attempts. These results show that jointly learning recovery decisions and feedback from task outcomes offers an effective approach to improving multi-step agent execution.

AI USE STATEMENT

In this work, we use generative AI tools to assist with manuscript writing and polishing. The authors review all AI-assisted content. We take full responsibility for the final content of this work, including its text, claims, and other materials produced with the aid of generative AI.

ETHICS STATEMENT

Our experiments use publicly available benchmarks for question answering, arithmetic reasoning, and web interaction, and do not recruit human participants. AgentReroute may make incorrect recovery decisions or provide misleading feedback, which can cause repeated actions, additional resource use, or inappropriate actions in real environments. The benchmark results do not establish reliability in irreversible or high-stakes tasks. Such applications require additional validation of critical actions and human oversight.

REPRODUCIBILITY STATEMENT

Sections 3.2–3.4 describe the structured recovery decisions, trajectory recovery, and training objective. Appendix A provides the training template and output parsing rules. Appendices B.1–B.5 describe the recovery procedure, task evaluation protocols, environment state restoration, model and training configurations, and baselines. Appendices C.1–C.3 define the behavior metrics and the protocols for cross-Generator evaluation and inference-cost accounting. Appendix D presents observed recovery trajectories from all three task families. Code and datasets are available in an anonymous repository at `https://anonymous.4open.science/r/AgentReroute-DCCA/`. We will also provide the same code and datasets as a compressed archive in the supplementary material in case the anonymous repository becomes temporarily unavailable.

REFERENCES

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. ReSearch: Learning to reason with search for LLMs via reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL `https://arxiv.org/abs/2503.19470v3`.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=KuPixIqPiq`.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. arXiv preprint arXiv:2501.12948, 2025. URL `https://arxiv.org/abs/2501.12948`.

Yunpeng Dong, Jingkai He, Shiqi Liu, Yuze Hou, Dong Du, Zhonghu Xu, Si Yu, Baochuan Yang, Yubin Xia, and Haibo Chen. DeltaBox: Scaling stateful AI agents with millisecond-level sandbox checkpoint/rollback. arXiv preprint arXiv:2605.22781, 2026. URL `https://arxiv.org/abs/2605.22781v2`.

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, 2025. URL `https://arxiv.org/abs/2504.11536`.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://arxiv.org/abs/2305.11738`.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625. International Committee on Computational Linguistics, 2020. doi: 10.18653/v1/2020.coling-main.580. URL `https://aclanthology.org/2020.coling-main.580/`.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.01798.

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. URL https://arxiv.org/abs/2503.09516v5.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D. Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M. Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=CjwERcAU7w.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. arXiv preprint arXiv:2501.05366, 2025a. URL https://arxiv.org/abs/2501.05366.

Xingzuo Li, Kehai Chen, Yunfei Long, Xuefeng Bai, Yong Xu, and Min Zhang. Generator-Assistant stepwise rollback framework for large language model agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 17683–17700, Suzhou, China, November 2025b. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.892. URL https://aclanthology.org/2025.emnlp-main.892/.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. arXiv preprint arXiv:2303.17651, 2023. URL https://arxiv.org/abs/2303.17651.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. arXiv preprint arXiv:2407.18219, 2024. URL https://arxiv.org/abs/2407.18219.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. AgentSquare: Automatic LLM agent search in modular design space. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/0ae94013da7cd459402fd77874e09ee3-Abstract-Conference.html.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9248–9274. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.620. URL https://aclanthology.org/2023.findings-emnlp.620/.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024. URL https://arxiv.org/abs/2402.03300.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. arXiv preprint arXiv:2303.11366, 2023. URL https://arxiv.org/abs/2303.11366.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-Searcher: Incentivizing the search capability in LLMs via reinforcement learning. arXiv preprint arXiv:2503.05592, 2025. URL https://arxiv.org/abs/2503.05592v2.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10014–10037. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.557. URL https://aclanthology.org/2023.acl-long.557/.

Qinzhuo Wu, Pengzhi Gao, Wei Liu, and Jian Luan. BacktrackAgent: Enhancing GUI agent with error detection and backtracking mechanism. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 4250–4272, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.212. URL https://aclanthology.org/2025.emnlp-main.212/.

Zhihui Xie, Jie Chen, Liyu Chen, Weichao Mao, Jingjing Xu, and Lingpeng Kong. Teaching language models to critique via reinforcement learning. arXiv preprint arXiv:2502.03492, 2025. URL https://arxiv.org/abs/2502.03492v2.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. arXiv preprint arXiv:2508.19828, 2025. URL https://arxiv.org/abs/2508.19828v5. Version 5, revised January 2026.

Ke Yang, Panpan Li, Zonghan Wu, Kejin Xu, Huaxi Huang, and Xiaoshui Huang. DART: Semantic recoverability for structured tool agents. arXiv preprint arXiv:2605.23311, 2026. URL https://arxiv.org/abs/2605.23311v1.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

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*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023a. URL https://arxiv.org/abs/2305.10601.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://arxiv.org/abs/2210.03629.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, Ran Xu, Phil Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. Retroformer: Retrospective large language agents with policy gradient optimization. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2308.02151v3.

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-R: Training language model agents to reflect via iterative self-training. arXiv preprint arXiv:2501.11425, 2025. URL https://arxiv.org/abs/2501.11425v3.

Haozhen Zhang, Tao Feng, and Jiaxuan You. Router-R1: Teaching LLMs multi-round routing and aggregation via reinforcement learning. arXiv preprint arXiv:2506.09033, 2025. URL https://arxiv.org/abs/2506.09033.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://arxiv.org/abs/2510.04618v3.

Zhisong Zhang, Tianqing Fang, Kaixin Ma, Wenhao Yu, Hongming Zhang, Haitao Mi, and Dong Yu. WebRollback: Enhancing web agents with explicit rollback mechanisms. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 187–197, Rabat, Morocco, March 2026b. Association for Computational Linguistics. doi: 10.18653/v1/2026.eacl-short.12. URL https://aclanthology.org/2026.eacl-short.12/.

Zhixin Zhang, Xinke Jiang, Zhibang Yang, Weixuan Xu, Guohong Qiu, Xu Chu, Junfeng Zhao, and Yasha Wang. LoongReflect: Boosting long-horizon reflection in search agents via global perspective distillation. arXiv preprint arXiv:2608.11967, 2026c. URL https://arxiv.org/abs/2608.11967v1.

Yu Zhuang, Kefei Chen, Yitong Duan, Shuxin Zheng, Jian Li, and Xu-Yao Zhang. AgentRewind: Recoverable execution for long-horizon LLM agents. arXiv preprint arXiv:2608.14380, 2026. URL https://arxiv.org/abs/2608.14380v1.

A Training Template and Recovery Interface

Table 4 gives the Rerouter's Game24 instruction template. The placeholder {numbers} is replaced with the four numbers of the current problem. The most recent Generator action and environment observation are appended inside <information> tags. During an episode, the Rerouter sees the current branch together with the available recovery history, as described in Section 3.2. It responds with either Continue or a rollback target with corrective feedback. The template makes the rollback target an action index: rolling back to Act N retains the actions before N and asks Generator to produce a different Act N.

Table 4: Game24 Rerouter training template. The boxed text follows the training prompt; the task numbers and the <information> block are filled in for each decision.

Game24 Rerouter training prompt

```
You are an expert reasoning supervisor for the "Game of 24" task.
Your goal is to monitor a Generator Agent as it attempts to reach
the target number 24 using basic arithmetic operations (+, -, *, /).
You will receive the Generator's latest action and the Environment's
observation wrapped in <information> tags.
After checking the latest information, output EXACTLY ONE of the
following decisions:
- If the trajectory is valid and promising, output:
<decision> Continue </decision>
- If you find an error and want the Generator to RE-DO a specific
  previous step N, output:
<decision> Rollback_to_N </decision>
<feedback> A brief correction for the Generator's next attempt </
    feedback>
(N must be a concrete step number. Never output the letter N or k.
`Rollback_to_N` means the Environment will keep all steps before
Act N, discard Act N and everything after it, and force the
Generator to try a DIFFERENT Act N based on your <feedback>.
The <feedback> should briefly explain the current mistake and give
one concrete correction for the Generator's next attempt. Do not
include long reasoning, multiple alternatives, or uncertainty.)
Question: Use numbers {numbers} to obtain 24.

<information>
Act 1> {generator action}
Obs 1> {environment observation}
</information>
```

A.1 Structured Output and Parsing

The Rerouter emits one decision block at each recovery check. A valid `Continue` decision advances Generator from the current active branch. A rollback decision must contain a concrete integer target in the form `Rollback_to_`N; the target denotes the first Generator action to regenerate. The executor retains actions $1, \ldots, N-1$, removes action N and its suffix, and passes the text inside `<feedback>` to Generator for the replacement action. Subsequent prompts contain the restored active trajectory.

B Reproducibility Details

B.1 Shared Recovery Protocol

The Generator and Rerouter alternate after each environment observation. Rerouter observes the task instruction, the active action–observation branch, and the recovery record. A `Continue` decision requests the next Generator action. An executed rollback first restores the Environment to the retained prefix and then requests a replacement action conditioned on the corrective feedback. Episode history used for learning contains all Rerouter responses, including decisions made before a rollback, whereas Generator is conditioned on the restored active branch and task-specific recovery context. An episode ends when the task-specific success condition is met, or the interaction budget is exhausted.

B.2 Task Protocols and Evaluation

WebShop. The Generator searches the catalog, opens a product page, selects the requested attributes, and submits a purchase. An action is followed by the page or environment response before Rerouter decides whether to continue or recover. We train on 750 tasks and evaluate on 500. The mean terminal environment score is reported as Score; SR is the fraction of tasks with a successful purchase. The raw terminal score supplies the training reward.

Game24. Each problem provides four numbers. Generator applies arithmetic operations to the remaining numbers and finally submits an expression for 24. We check an action against the Environment's resulting multiset; a successful terminal expression must use the supplied numbers and equal 24. We train on 250 problems and evaluate on 500, reporting the fraction solved. The training reward is the binary success indicator. The active branch contains at most four Generator actions. After rolling back to Act N, the Environment reconstructs the state after Act $N-1$ before the Generator proposes a replacement.

Question answering. The Rerouter is trained on 1,000 MuSiQue questions, and we evaluate the same trained policy without further training on HotpotQA, 2WikiMultihopQA, MuSiQue, and Bamboogle (500, 500, 500, and 125 questions, respectively). Generator can issue a search query, inspect retrieved passages, and submit a final answer. We report answer exact match and token-level F1; training uses final-answer F1. A rollback restores the earlier search or answer position, so a revised query does not inherit the discarded suffix as an active branch. The retrieval evidence in the 2WikiMultihopQA example in Table 10 shows this distinction.

B.3 Environment State Restoration

Rollback changes both the textual context and the task state. For Game24, the executor replays the retained arithmetic actions from the four input numbers and reconstructs the multiset available immediately before action N. The discarded arithmetic operations are excluded from the replacement prompt. For question answering, the executor retains the earlier query–evidence prefix and removes later queries, retrieved passages, and answer attempts from the active branch. For WebShop, the executor reconstructs the Environment state associated with the retained interaction prefix before issuing the replacement action. In all three task families, the removed suffix remains visible to Rerouter through the recovery record but is not treated as active Generator history. This separation prevents an abandoned observation from continuing to define the Environment state after rollback.

B.4 Models and Training Configuration

Table 5 records the model assignments used in the reported task families. Generator weights remain frozen. For each task–Generator pair, Rerouter starts from Qwen3-4B-Instruct-2507 and is trained with group relative policy optimization. The return comes from the terminal task outcome specified in Appendix B.2; responses from one complete episode share that episode's group-mean-centered advantage. Rerouter rollouts use temperature 1.0 during training and greedy decoding during validation and testing. The Generator temperature is 0.1 across task families.

Table 5: Model assignments and outcome signals. Each task–Generator pair has a separately trained Rerouter.

Task family	Frozen Generator	Training outcome
WebShop	Qwen3-8B or Llama-3.2-3B-Instruct	Raw terminal environment score
Game24	Qwen3-8B or Llama-3.2-3B-Instruct	Binary task success
Question answering	Qwen3-8B	Final-answer token F1

B.5 Baselines and Combination Protocol

For WebShop and Game24, Act-only executes Generator without a recovery controller. ReAct retains the standard action–observation interaction, Reflexion adds between-attempt reflection, and GA-

Rollback supplies a prompt-based recovery controller. AgentReroute + Reflexion keeps the trained Rerouter active within each attempt and applies Reflexion only between attempts; the second attempt is therefore conditioned on the reflection while retaining the same within-attempt recovery mechanism. For question answering, the Base model has no retrieval, Naive RAG retrieves once before answering, and Iter-RetGen, IRCoT, and Search-o1 are multi-step retrieval baselines. AgentSquare and ACE are included as general agent optimization baselines. All comparisons use final task outcomes rather than intermediate recovery judgments as evaluation targets.

C Additional Quantitative Details

C.1 Recovery-Behavior Metrics

We identify an executed rollback only when the Environment confirms restoration to the requested target. Rollback distance is the number of discarded Generator actions. To prevent episodes with repeated rollback loops from dominating Figure 3(a), distance shares are computed within each episode and then averaged across episodes containing at least one executed rollback. First-rollback depth is the active-branch length immediately before the first executed rollback. State recurrence in Figure 3(b) compares the Environment state after the first valid replacement with the state produced by the discarded action; episodes without two parseable states are excluded from this conditional measure. Table 6 reports the underlying counts and summaries. The three conditions use independently sampled trajectories, so the comparison is descriptive.

Table 6: Numeric details for Figure 3. Distance columns are episode-balanced shares (%); recurrence gives the count over valid replacements and the corresponding rate.

Condition	RB episodes	RB/task	Dist. 1	Dist. 2	Dist. $\geq$3	First depth	Recurrence
Full	442/500	2.50	59.5	38.9	1.6	2.22	94/429 (21.9%)
w/o Feedback	442/500	4.31	65.5	33.4	1.1	2.27	409/439 (93.2%)
w/o RL Training	483/500	4.80	63.4	36.0	0.7	2.47	333/479 (69.5%)

C.2 Cross-Generator Protocol

The cross-Generator evaluation uses the same 500 Game24 test problems for all six pairings. The prompt-only Rerouter and each trained Rerouter remain fixed while the inference-time Generator is changed. Thus, an off-diagonal entry in Table 7 measures direct reuse without additional Rerouter training under the new Generator. The corresponding training curves use a separate fixed set of Game24 problems and report measured checkpoint outcomes.

Table 7: Cross-Generator reuse on Game24. π_0 is the prompt-only Rerouter; π_Q and π_L are trained with Qwen3-8B and Llama-3.2-3B, respectively. Entries are success rates (%).

Rerouter	Training Generator	Qwen3-8B	Llama-3.2-3B
π_0	None	15.6	9.0
π_Q	Qwen3-8B	77.8	53.8
π_L	Llama-3.2-3B	57.6	52.8

C.3 Inference-Cost Accounting

The inference-cost comparison in Table 3 is computed over the 500-task Game24 evaluation with Llama-3.2-3B-Instruct fixed as Generator. Gen. aggregates tokens consumed by Generator calls; Aux. aggregates tokens consumed by the recovery or reflection module, and Time measures end-to-end wall-clock latency from task initialization to termination. All values are averaged per task under the execution protocol used for the corresponding method.

D REPRESENTATIVE RECOVERY CASES

Tables 8–10 show one successful run from each task family. We preserve the decision sequence and the action or evidence relevant to the correction; long environment observations and feedback are shortened for readability.

Table 8: Game24: replacing an unproductive second operation while retaining the first. Input numbers: 4, 4, 4, 11.

Stage	Observed trajectory and recovery
State	Act 1: $11 - 4 = 7$, leaving $4, 4, 7$. Act 2: $7 - 4 = 3$, leaving $3, 4$.
Decision	Rerouter selects `Rollback_to_2` and advises multiplying 7 by 4 before subtracting the remaining 4.
Result	The replacement Act 2 computes $4 \times 7 = 28$. Act 3 computes $28 - 4 = 24$, and the final expression $(4 \times (11 - 4)) - 4 = 24$ succeeds.

Table 9: WebShop: correcting an invalid color selection. The request asks for a large brown slim-fit active shirt below $40.

Stage	Observed trajectory and recovery
State	Generator searches for the requested shirt, opens product `B07H9WCYGF` ($31.96–$31.99), and selects size `large`. The product page lists `wolf brown` as a color option, but Act 4 attempts `click[brown]` and receives `Invalid action!`
Decision	Rerouter selects `Rollback_to_4` and identifies `wolf brown` as the available color option.
Result	The replacement Act 4 uses `click[wolf brown]`, which the Environment accepts. Generator then selects `Buy Now`; the purchase succeeds with a terminal score of 1.0.

Table 10: 2WikiMultihopQA: separating the film-director and birthplace hops. Question: "Where was the director of film The Countess (Film) born?"

Stage	Observed trajectory and recovery
State	Act 1 searches for `Director of The Countess (Film) birthplace`. The retrieved passages include an unrelated director, Edmond T. Gréville, alongside evidence that Julie Delpy directed the film.
Decision	Rerouter selects `Rollback_to_1`. Its feedback points to Julie Delpy and directs Generator to establish the director before searching for the birthplace.
Result	The revised Act 1 confirms Julie Delpy as the director. Act 2 searches for `Julie Delpy birthplace` and retrieves evidence that she was born in Paris. Act 3 answers `Paris`; the recorded answer has EM $= 1$ and F1 $= 1$.