AGENT REINFORCEMENT LEARNING VIA PIVOTAL-AWARE SELF-FEEDBACK RETRY

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents have shown strong decision-making capabilities in long-horizon interactive tasks, yet they still struggle to effectively leverage failed trajectories: full retries incur high interaction costs, while experience retrieval tends to dilute critical experience signals. To address this, we propose PivoARL, a self-feedback retry framework for experience exploitation in LLM agents. PivoARL identifies the pivotal erroneous turn through structured reflection and performs local retry only from the corresponding pivotal state, thereby reusing the correct prefix and reducing redundant interactions. From an information-gain perspective, we further show that pivotal retry concentrates useful experience signals near the error boundary, mitigating the signal dilution caused by state-agnostic experience utilization. Based on this insight, we design a pivotal-aware credit assignment mechanism that rewards correct prefixes while isolating erroneous suffixes, and optimize reflection quality through implicit reflection returns. We conduct a systematic evaluation on 4 agent tasks and 7 search-based QA benchmarks. Results show that PivoARL achieves significant improvements on Pass@2/3 across all tasks, with an average gain of about 11.5% over MetaRL. Moreover, benefiting from contrastive preference signals induced by pivotal turns, PivoARL also improves Pass@1 over the default RL baseline on over 80% of the evaluated tasks. On Minesweeper, PivoARL improves over GiGPO by more than 45% on average across Suc@1/2/3, while reducing training interaction turns by about 35% over MetaRL on ALFWorld.

1 INTRODUCTION

Large language models (LLMs) have evolved from static question-answering systems into versatile *agents* that perceive, reason, and act across multi-turn interactions with external environments (Yao et al., 2023; Shinn et al., 2023; Guo et al., 2025). These agents navigate simulated households (Shridhar et al., 2021), plan multi-step web browsing sessions (Yao et al., 2022; Chai et al., 2025), code development (Guo et al., 2026), and solve complex tasks that demand long-horizon planning and adaptive decision-making. However, training efficiency remains a bottleneck due to high interaction costs and sparse rewards (Zhang et al., 2026; Cai et al., 2025). A fundamental challenge persists: *how to effectively mine experiences from past trajectories to facilitate robust learning?*

Despite its potential in long-horizon agent tasks (Wang et al., 2025), experience-based reinforcement learning (RL) faces dual challenges (Wang et al., 2026c). First, there is a trade-off between precision and efficiency in experience utilization: static library-based methods (e.g., SkillsRL (Xia et al., 2026)) struggle to adapt to dynamic task evolution, while dynamic paradigms are often susceptible to interference when processing large-scale experiences. Second, credit assignment remains a major challenge in long-horizon agent tasks. Sparse final feedback from failed trajectories makes it difficult to localize the exact erroneous step (Jiang et al., 2026), causing negative advantages to be mistakenly assigned to correct prefix decisions. This weakens the learning of useful experiences and may lead to training instability or even capability degradation (Li et al., 2026; Shi et al., 2026a).

In this study, we propose PivoARL, which treats the pivotal turn as both a retry and credit boundary. Unlike prior pivotal retry that mainly determines where to regenerate (Shi et al., 2026b), PivoARL further propagates successful retry signals to the reused correct prefix while isolating the replaced erroneous continuation. To optimize the learning process, we design a cross-episode prefix credit

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.

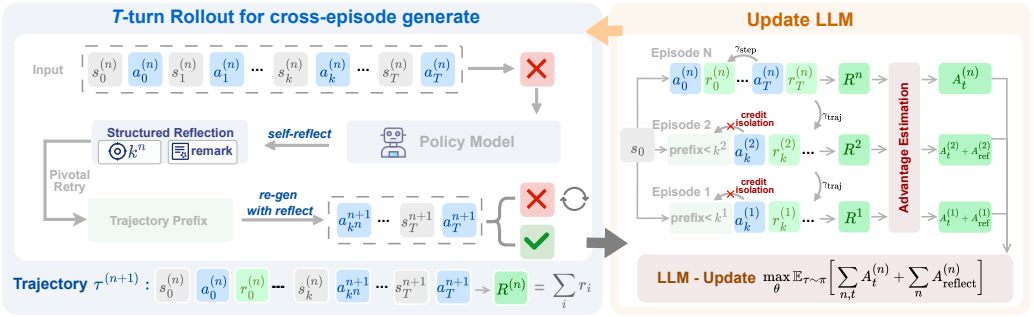


Figure 1: **The Pivotal-Aware Self-Feedback Retry (PivoARL) framework**. The agent identifies pivotal errors via structured self-reflection, retries from the pivotal state with the correct prefix reused, and updates the LLM through pivotal-isolated cross-episode credit assignment.

assignment mechanism and implicit reflection returns. These components enable the agent to deeply mine effective signals from failed trajectories and ultimately transform them into reflection-guided, high-quality successful experiences. From an information-gain perspective, we further demonstrate that pivotal retries concentrate useful experience signals near the error boundary, effectively mitigating the signal dilution commonly found in global reflections.

Experimental results show that PivoARL brings consistent gains across four agent environments and seven search-based QA benchmarks. On agent tasks, PivoARL improves over GiGPO by about 10.3%, 15.3%, and 14.7% on Pass@1/2/3, respectively; on search-based QA tasks, it improves over GRPO by about 22.6%. Meanwhile, benefiting from the efficient reuse mechanism of pivotal retry, PivoARL reduces training interaction cost by about 13.5% on average compared with MetaRL. We further conduct detailed ablation studies to quantify the contribution of each component, and perform additional analyses to examine the test-time and training-time scaling behavior of PivoARL.

Our contributions are as follows:

- **Pivotal-Aware Cross-Episode Learning.** We treat the pivotal state as both a retry and credit boundary, enabling successful retry signals to reinforce the reused correct prefix while preventing reward leakage into the erroneous continuation.
- **Information Gain Analysis.** We analyze experience learning from an information-gain perspective, showing that global reflections or retrieved experiences are easily diluted in long trajectories, while pivotal retry concentrates useful guidance around the critical error boundary.
- **State-of-the-Art Performance.** PivoARL improves Pass@k across four agent environments and seven search-based QA benchmarks, achieving about 16% average relative gain over default RL baselines, while demonstrating stronger test-time and training-time scaling.

2 Preliminary

2.1 Notation and Problem Setup

We formulate LLM agent interaction as a Markov decision process (Jiang et al., 2026; Cai & Sugiyama, 2026) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$. Given a task $x \sim D$, the agent operates over T steps: at step t, it observes state $\boldsymbol{s}_t \in \mathcal{S}$, generates action $\boldsymbol{a}_t \in \mathcal{A}$ via policy $\pi_\theta(\boldsymbol{a}_t \mid \boldsymbol{s}_t)$. The environment then provides a scalar reward $r_t \in \mathbb{R}$, observation o_t and transitions to $\boldsymbol{s}_{t+1}$. For each trajectory $\boldsymbol{\tau} = \{(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{r}_0), \ldots, (\boldsymbol{s}_{T-1}, \boldsymbol{a}_{T-1}, \boldsymbol{r}_{T-1})\}$, we utilize the total return $R(\tau) = \sum_{t=0}^{T-1} r_t$ as a episode-level reward.

Self feedback with experience cycle. Beyond scalar rewards, the historical trajectory $\boldsymbol{\tau}_0$ provides rich feedback at each step (e.g., error messages, state descriptions), revealing *why* the agent succeeded or failed (Xiao et al., 2026). Given the interaction context $\boldsymbol{c} = \sum_t^{T-1}\{(a_t, o_t)\}$ from an initial

trajectory, the agent constructs an augmented context to extract the past experience information:

$$\textbf{remark} = \pi_\theta\big(\cdot\,|x,\ \{\boldsymbol{c}\}\big) \tag{1}$$

Where **remark** represents the experience gained from the previous attempt, utilized in the next attempt (Shi et al., 2026b) as a new enhanced task prompt $\boldsymbol{x}^+$. This enables the model to leverage past experience for optimized subsequent attempts.

2.2 Agentic Reinforcement Learning

Agentic Reinforcement Learning (ARL) typically adopts policy-gradient methods (Schulman et al., 2017) to optimize the agent policy π_θ. We formulate the agentic RL training objective as:

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D},\tau\sim\pi_\theta(\cdot|x)}\left[\sum_{t=0}^{T-1} A_t\right] - \beta\mathbb{D}_{\text{KL}}\left[\pi_\theta(\cdot)\|\pi_{\text{ref}}(\cdot)\right] \tag{2}$$

where π_{ref} is the reference LLM, and $\mathbb{D}_{\text{KL}}$ and β denote the KL divergence and its coefficient, respectively. A_t is the step-wise advantage, typically computed from both step-level and episode-level rewards within the group.

3 Method

3.1 Overview: The PivoARL Framework

The previous cross-episode paradigm (Jiang et al., 2026) treats each retry as a full restart from s_0, discarding the correct prefix of the failed trajectory. We propose **Pivotal-Aware Self-Feedback**, which reuses the correct prefix and restarts from the pivotal error.

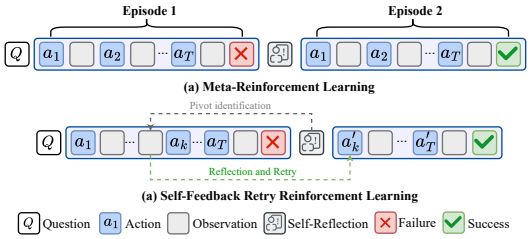


Figure 2: Comparison between MetaRL and Pivotal-Aware Self-Feedback Retry RL.

Pivotal Retry Training Framework. In the training of PivoARL, each trial consists of N episodes sequentially generated by the agent:

$$\mathcal{T} = (\tau^{(0)}, \tau^{(1)}, \ldots, \tau^{(N-1)}),\ n \in [0, N-1] \tag{3}$$

The rollout process terminates at n if τ^n is successful. Otherwise, the agent initiates a new episode $\tau^{(n+1)}$. Unlike prior work relying on past experiences for a full restart, our agent reflects to identify pivotal decision points, enabling partial trajectory reuse.

Structured Reflection and Pivotal Retry. After a failed episode $\tau^{(n)}$ of $T^{(n)}$ turns, the agent generates a structured reflection containing a textual remark and a *pivotal turn index* $k^{(n)} \in [0, T^{(n)} - 1]$ that identifies the earliest erroneous turn. The remark provides experience guidance injected into the original state during retry:

$$\Delta^{(n)}_{\text{reflect}} \sim \pi_\theta\big(\cdot \mid x, \boldsymbol{c}^{(n)}\big) = \big(k^{(n)}; \text{remark}^{(n)}\big). \tag{4}$$

Given the pivotal turn $k^{(n)}$, the next episode $\tau^{(n+1)}$ reuses the correct prefix of $\tau^{(n)}$, regenerating actions only from $k^{(n)}$:

$$\tau^{(n+1)} = \Big(\underbrace{s_0^{(n)}, a_0^{(n)}, \ldots, s_{k^*}^{(n)}}_{\text{reused prefix (not re-optimized)}}, \underbrace{a_{k^*}^{(n+1)}, s_{k^*+1}^{(n+1)}, \ldots, a_{T^{(n+1)}-1}^{(n+1)}}_{\text{regenerated from } k^* \text{ (where } k^*=k^{(n)})} \Big). \tag{5}$$

Where $T^{(n+1)}$ is the length of the new episode. This reduces the generation overhead for each retry from $T^{(n+1)}$ to $T^{(n+1)} - k^{(n)}$ tokens, while preserving the correct decision history as context. Note that the reused prefix is preserved only as interaction context and is not re-sampled or re-optimized in episode $n+1$; its actions are updated only at their original occurrence in episode n.

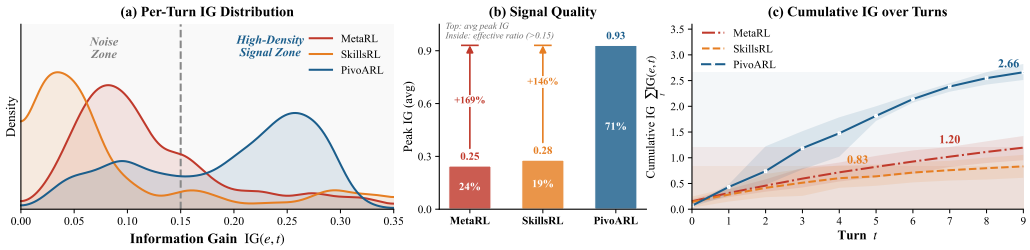


Figure 3: Information gain (IG) analysis of experience guidance. (a) ***Per-turn IG density***: our method concentrates IG in high-signal turns. (b) ***Signal Quality***: Peak IG and percentage of high-IG turns. (c) ***Cumulative IG over turns***: showing a characteristic leap at turn k for our method.

3.2 Pivotal-Aware Credit Assignment

Credit Assignment with Pivotal Isolation. Pivotal retry creates a cross-episode credit boundary, where successful retry returns should reinforce the reused prefix without rewarding the erroneous continuation. We decompose $G_t^{(n)}$ into intra-episode returns $g_t^{(n)}$ and inter-episode returns:

$$g_t^{(n)} = \sum_{l=t}^{T^{(n)}-1} \gamma_{\text{step}}^{l-t}\, r_l^{(n)}, \qquad G_t^{(n)} = g_t^{(n)} + \sum_{m=n+1}^{N-1} \gamma_{\text{traj}}^{m-n}\, g_0^{(m)}. \tag{6}$$

Where γ_{step} and γ_{traj} are the intra- and inter-episode discount factors, respectively.

However, naively applying this in the pivotal setting lets erroneous steps ($t \geq k^{*(n)}$) receive positive returns from the successful retry, rewarding the very mistakes that caused failure (Yu et al., 2026). We therefore treat $k^{*(n)}$ as an explicit credit boundary and introduce pivotal credit isolation:

$$G_t^{(n)} = \begin{cases} \displaystyle\sum_{l=t}^{k^{(n)}-1} \gamma_{\text{step}}^{l-t} r_l^{(n)} + \gamma_{\text{step}}^{k^{(n)}-t} \gamma_{\text{traj}} G_{k^{(n)}}^{(n+1)}, & t < k^{(n)}, \\ \displaystyle\sum_{l=t}^{T^{(n)}-1} \gamma_{\text{step}}^{l-t} r_l^{(n)}, & t \geq k^{(n)}. \end{cases} \tag{7}$$

Here, $G_{k^{(n)}}^{(n+1)}$ denotes the return of the regenerated suffix from the pivotal state, with the reused prefix excluded from repeated credit. We compute $G_t^{(n)}$ backward across episodes, preventing future retry credit from leaking into the erroneous suffix.

Reflection Reward through Implicit Returns. To co-optimize reflection generation and action retries, we define the reflection reward $r_{\text{reflect}}^{(n)}$ as the discounted return of the subsequent retry episode:

$$r_{\text{reflect}}^{(n)} = \mathbb{I}(\tau^{(n+1)} \text{ succeeds}) \sum_{l=k^{(n)}}^{T^{(n+1)}-1} \gamma_{\text{step}}^{l-k^{(n)}}\, r_l^{(n+1)}. \tag{8}$$

This reward incentivizes precise failure localization by coupling reflection quality with retry success and penalizing redundancy via γ_{step}. Moreover, the reward is contingent upon correct formatting to ensure structural validity.

Optimization. Given the per-action cross-episode return $G_t^{(n)}$ defined above, we define the gradient objective as follows:

$$\nabla_\theta \mathcal{L}_{\text{RL}}(\theta) = \mathbb{E}_{\boldsymbol{\tau}\sim\pi_\theta}\Big[\sum_{n,t} \nabla_\theta \log \pi_\theta\big(a_t^{(n)} \mid s_t^{(n)}\big) A_t^{(n)} + \sum_{n} \nabla_\theta \log \pi_\theta\big(\cdot \mid x, \boldsymbol{c}^n\big) A_{\text{reflect}}^{(n)} \Big]. \tag{9}$$

Where $A_t^{(n)}$ is the advantage estimate from $G_t^{(n)}$, and $A_{\text{reflect}}^{(n)}$ is derived via group-wise normalization of the reflection rewards $r_{\text{reflect}}^{(n)}$. The framework is compatible with widely used optimizers such as GRPO and GiGPO (Feng et al., 2025). See Appendix C.3 for the detailed advantage computation.

Notably, pivotal retry also improves the model's first-attempt success ability (See in Appendix E.3).

3.3 Concentrated Information Gain Analysis

We analyze PivoARL through an information-gain lens to provide an explanatory view of pivotal retry. Compared with global guidance, pivotal retry can concentrate useful signals near the error boundary, as empirically illustrated in Fig. 3, thereby improving the efficiency of experience utilization. See Appendix E.2 for details of the empirical IG analysis.

Per-Turn Information Gain. Given a trajectory τ and experience guidance $\textbf{remark}^{(n)}$, we define the *per-turn information gain*:

$$\mathrm{IG}(e,t) = \log \pi_\theta(a_t^* \mid s_t, e) - \log \pi_\theta(a_t^* \mid s_t). \tag{10}$$

High $\mathrm{IG}(e,t)$ indicates actionable guidance at turn t; near-zero values indicate noise.

Our analysis suggests that existing experience utilization paradigms may suffer from low *effective signal density* (Wang et al., 2026b), i.e., few turns receive non-negligible IG.

Proposition 1 (IG dilution under bounded global guidance)**.** *Let $\rho(e) = \frac{1}{T}|t : \mathrm{IG}(e,t) > \delta|$ denote the effective signal density of guidance e. Under a bounded-influence assumption, a fixed global reflection Δ reused across all turns has effective signal density $\rho(\Delta) = O(1/T)$ as T grows. For finite-memory retrieval, we separately analyze the sparse state-coverage effect in Appendix E.1.*

Concentrated Information Gain via Pivotal Retry. By identifying the error boundary k^* and applying state-specific guidance at the pivotal turn, our mechanism concentrates information gain:

$$\mathrm{IG}(e,t) \approx \begin{cases} \mathrm{IG}_{\mathrm{high}}, & t \geq k^*, \\ 0, & t < k^*, \end{cases} \tag{11}$$

where $\mathrm{IG}_{\mathrm{high}} \gg \frac{1}{T}\sum_t \mathrm{IG}(\Delta, t)$. This provides an intuitive explanation for why pivotal retry may yield more concentrated guidance: the correct prefix requires no guidance, while the error suffix is regenerated from k^*. As training proceeds, k^* moves later, yielding more precise experience signals.

4 Experiments

4.1 Experimental setup

Environment. We evaluate PivoARL on four agent exploration environments (Sciworld (Wang et al., 2022), Minesweeper (Li et al., 2024), WebShop (Yao et al., 2022), and ALFWorld (Shridhar et al., 2021)) and multi-step search tasks, where we use a constructed Wiki knowledge base as the retrieval source. And we evaluate the multi-turn search performance on single-hop QA datasets (NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2023)) and multi-hop QA datasets (HotpotQA (Yang et al., 2018), 2Wiki (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023)). Details are provided in Appendix C.

Baselines. We compare PivoARL with six categories of competitive baselines. 1) ***Closed-source LLMs***, including GPT-4o and Gemini-2.5-Pro. 2) ***Prompt-based Agentic Methods***, including ReAct (Yao et al., 2023) and Reflexion (Shinn et al., 2023). 3) ***RL-based Methods***, including group-based online RL algorithms such as GRPO, DAPO and GiGPO, as well as the clipping-based RL algorithm GSPO (Zheng et al., 2025a). 4) ***Memory-Augmented RL-based Methods***, including Mem0+GRPO (Chhikara et al., 2025), SimpleMem+GRPO (Liu et al., 2026), and SkillRL (Xia et al., 2026). 5) ***Reflect-Retry RL-based Methods***, including R3L (Shi et al., 2026b) and MetaRL (Jiang et al., 2026). 6) ***Classic Agentic Search Methods***, including Search-R1 (Jin et al., 2025), ZeroSearch (Sun et al., 2025), and StepSearch (Zheng et al., 2025b).

Training details. We use Qwen3-4B as the base model and adopt GiGPO as the default optimization algorithm across all experiments. For PivoARL training, we sample $N = 3$ episodes and set the group size to 8 for agent tasks and group size to 4 for search tasks. All experiments are conducted on a cluster with 8 NVIDIA H100-80GB GPUs. Detailed training settings are provided in Appendix C.

Table 1: Main results across four agent benchmarks with Qwen3-4B. **Suc@1/2/3**: success rate in 1, 2 or 3 rollouts (%); **Avg.** denotes the mean of all success metrics across environments. The best results and second best results are highlighted in red and blue, respectively.

Methods	ALFWorld			WebShop			Minesweeper			SciWorld			Avg.
	Suc@1	Suc@2	Suc@3	Suc@1	Suc@2	Suc@3	Suc@1	Suc@2	Suc@3	Suc@1	Suc@2	Suc@3	
Closed-source LLMs													
GPT-4o	42.5	48.3	50.8	33.6	37.8	39.5	22.5	26.4	26.7	32.8	42.1	47.6	37.6
Gemini-2.5-Pro	54.3	61.2	64.8	37.6	39.7	40.4	24.2	29.6	31.2	42.9	55.4	60.9	45.2
Prompt-based Agentic Methods													
ReAct[ICLR 23]	17.1	21.8	27.2	2.30	5.40	7.5	3.12	3.96	5.83	9.4	12.5	12.5	10.7
Reflexion[NeurIPS 23]	17.9	28.1	29.7	1.55	3.18	5.1	3.12	5.46	7.49	9.3	14.7	17.9	11.9
RL-based Methods													
GRPO	76.4	80.3	82.6	61.7	68.8	72.7	39.1	42.1	45.3	75.8	82.8	86.7	67.9
DAPO[NeurIPS 25]	82.0	89.8	91.2	64.1	69.5	75.0	46.0	50.0	51.6	83.5	94.5	98.4	74.6
GiGPO[NeurIPS 25]	85.9	87.5	89.8	67.9	80.5	82.6	42.2	46.1	48.4	78.9	86.7	89.8	73.9
GSPO[arXiv 25]	85.9	91.4	91.4	61.7	67.2	68.6	46.1	57.0	57.8	89.8	94.5	96.0	75.6
Memory-Augmented RL-based Methods													
Mem0+GRPO[ECAI 25]	82.8	84.7	85.2	63.1	69.5	74.8	44.5	46.8	49.2	79.6	91.4	93.2	72.1
SimpleMem+GRPO[ICML 26]	87.5	89.0	90.6	66.4	74.2	78.1	43.0	50.0	51.5	81.7	90.2	92.6	74.6
SKillRL[NeurIPS 26]	83.6	91.4	92.5	**68.2**	72.6	74.2	**50.7**	56.3	57.9	82.1	89.0	89.8	75.7
Reflect-Retry RL-based Methods													
R3L[arXiv 26]	79.2	82.1	84.9	60.7	71.9	73.8	39.2	51.7	57.1	73.4	83.4	88.9	70.5
MetaRL[ICLR 26]	82.0	85.9	86.7	63.2	71.3	75.2	39.8	54.9	61.5	94.5	97.6	98.4	75.9
PivoARL (Ours)	**89.8**	**93.4**	**94.5**	66.7	**81.5**	**83.8**	48.4	**71.8**	**78.1**	**98.4**	**100**	**100**	**83.9**

Table 2: Pass@3 performance on search-augmented QA tasks with Qwen3-4B. Models are trained on NQ and HotpotQA. † and ⋆ indicate in-domain and out-of-domain datasets, respectively.

Method	Single-Hop QA			Multi-Hop QA				Avg.
	NQ†	TriviaQA⋆	PopQA⋆	HotpotQA†	2Wiki⋆	MuSiQue⋆	Bamboogle⋆	
Classic Agentic Search Methods								
Search-R1[COLM 25]	48.3	63.6	48.9	47.3	52.4	22.9	46.4	47.1
ZeroSearch	51.2	63.1	48.3	48.1	47.3	23.1	46.4	46.8
StepSearch[EMNLP 25]	49.3	65.8	47.9	48.8	50.0	22.4	51.2	47.9
RL-based Methods								
GRPO	49.7	63.7	50.0	47.3	46.4	20.7	47.2	46.4
GiGPO[NeurIPS 25]	49.7	67.2	49.1	48.2	47.9	20.0	48.0	47.2
GSPO[arXiv 25]	48.8	64.8	50.6	45.9	43.0	18.5	40.8	44.6
Reflect-Retry RL-based Methods								
MetaRL[ICLR 26]	51.7	67.8	49.8	**54.6**	55.2	24.4	53.6	51.0
PivoARL (Ours)	**59.8**	**71.6**	**59.4**	53.0	**64.0**	**31.1**	**59.2**	**56.9**

4.2 Main Result

We evaluate all methods on four agent benchmarks and seven search-based QA benchmarks. Table 1 summarizes the final performance on agent tasks, while Table 2 reports comparisons with search-related methods. Figure 4 shows the training curves across representative agent and search tasks.

Comparison on Agent Tasks. Across all four agent environments, PivoARL consistently outperforms other RL training methods and Reflect-Retry baselines on all Succ@2/3 metrics. Overall, our method achieves an average 10.5% improvement over the state-of-the-art MetaRL. In the Minesweeper environment, PivoARL surpasses GiGPO and MetaRL by 45.1% and 27.0%, respectively. Moreover, benefiting from contrastive advantages at pivotal turns, PivoARL also obtains an average 6.0% gain on most Succ@1 metrics, demonstrating its superior effectiveness.

Comparison on Search Tasks. Across seven single-hop and multi-hop retrieval tasks, PivoARL achieves the best Pass@3 performance on 6/7 benchmarks, and obtains the second-best result on HotpotQA. Compared with the GRPO baseline, PivoARL obtains a 22.6% relative improvement. Compared with classic agentic search methods, PivoARL improves performance by 20.4% on

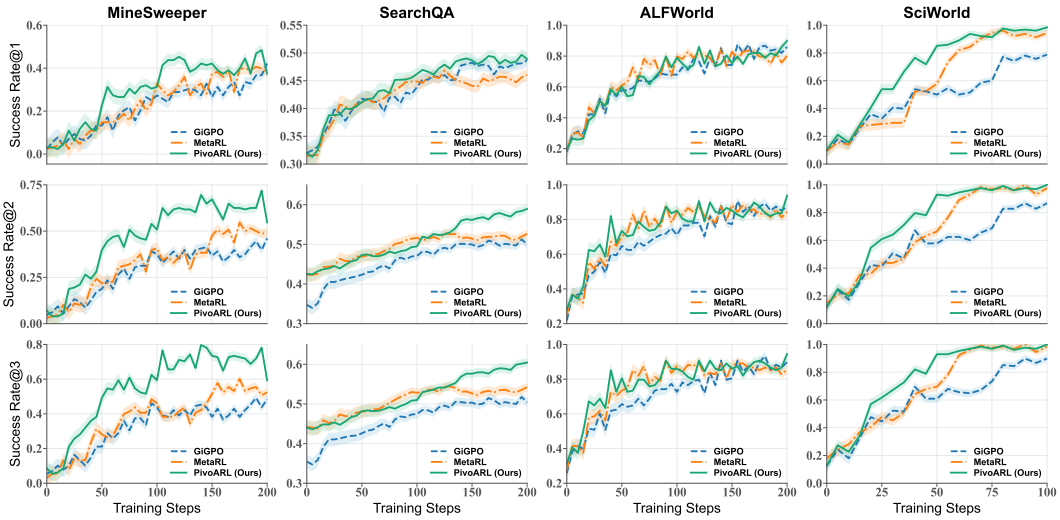


Figure 4: Training curves across representative agent and search benchmarks. The top, middle, and bottom rows show Success@1, Success@2, and Success@3, respectively, plotted against training steps. We compare GiGPO (blue, dashed), MetaRL (orange, dashed), and PivoARL (green, solid).

average. Compared with the reflect-retry baseline MetaRL, PivoARL further achieves an 11.6% relative improvement. Detailed Pass@1 and Pass@2 results are provided in Appendix D.1.

Comparison with Training Dynamics. We present the Pass@1/2/3 training curves of PivoARL, GiGPO, and MetaRL. Overall, PivoARL shows more stable training advantages across different tasks. On Minesweeper, PivoARL significantly accelerates learning convergence and consistently improves all three metrics. On search tasks, PivoARL also brings moderate gains in Pass@1, with more pronounced improvements on Pass@2/3, indicating stronger advantages in multi-attempt settings.

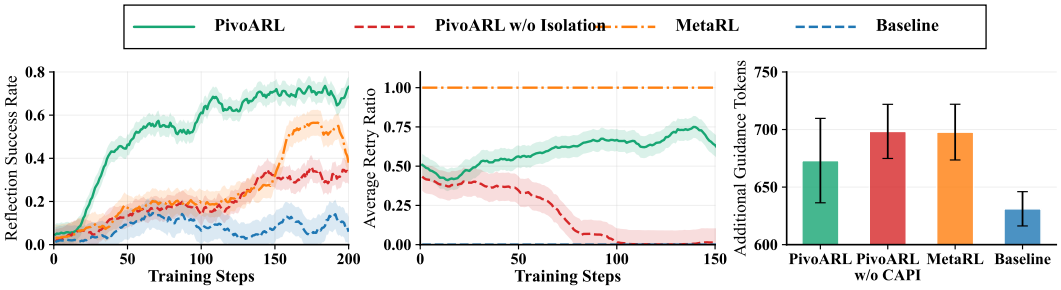
4.3 Ablation Studies

Ablation on Component. To analyze the contribution of each component in PivoARL, we conduct the following ablation experiments: a) ***w/o Reflection***: removing reflection experience injection; b) ***w/o Pivotal***: removing pivotal turn localization and instead performing a full retry from the initial state; and c) ***w/o Isolation***: removing the pivotal-isolated credit assignment mechanism. The results are shown in Table 3. Without reflection experience, the model performs similarly to vanilla GiGPO, indicating that reflection injection is an important source of improved exploration efficiency. Removing the pivotal mechanism reduces our method to a full-retry paradigm similar to MetaRL. In contrast, credit isolation plays a key role in pivotal retry by preventing gradient updates from mistakenly penalizing the correct prefix, bringing an average improvement of 10.1 percentage points over the w/o Isolation, with a particularly large gain of 23.5 percentage points on Minesweeper.

Table 3: Ablation study of PivoARL components on three agent environments. We report Suc@3 (%).

Method	ALFWorld	SciWorld	Minesweeper
GiGPO	89.8	89.8	48.4
PivoARL			
w/o *Reflection*	87.5$^{-2.3}$	86.3$^{-3.5}$	50.5$^{+2.1}$
w/o *Pivotal*	86.7$^{-3.1}$	98.4$^{+8.6}$	61.5$^{+13.1}$
w/o *Isolation*	91.8$^{+2.0}$	95.9$^{+6.1}$	54.6$^{+6.2}$
PivoARL	**94.5**$^{+4.7}$	**100**$^{+10.2}$	**78.1**$^{+29.7}$

Ablation on Credit Assignment. As shown in Figure 6, PivoARL demonstrates superior exploration efficiency and prompt economy over baselines. In terms of reflection capability (Fig. 6a), PivoARL achieves a peak success rate of 83.8%, outperforming MetaRL 58.9% and the baseline. When evaluating retry efficiency (Fig. 6b), our method maintains a competitive average retry ratio of 0.44, requiring significantly fewer redundant interactions than MetaRL (constant 1.0). Notably, removing the CAPI mechanism (w/o CAPI) causes a noticeable drop in success rate and an increase



(a) Reflection Success Rate (b) Reflection Retry Efficiency (c) Additional Guidance Tokens

Figure 6: Performance and efficiency comparison on Minesweeper. We evaluate PivoARL, its ablated variant (w/o Credit Assignment with Pivotal Isolation (CAPI)), MetaRL, and the Baseline across three metrics: (a) Reflection Success Rate, (b) Average Retry Ratio, (c) Additional Guidance Tokens.

in retry frequency, confirming CAPI's necessity. Furthermore, analysis of token overhead (Fig. 6c) reveals that PivoARL consumes only 716 additional guidance tokens on average—comparable to the ablated variant and substantially fewer than MetaRL.

4.4 Analysis

Analysis of Preference Group Density. To understand why pivotal retry also improves Pass@1, we count the preference groups produced by different retry strategies. A preference group refers to trajectories that share the same state but receive different rewards. As shown in Figure 7a, reflective retry consistently produces the highest number of preference groups throughout the training process, achieving an overall 48.4% increase over MetaRL, mainly concentrated in the middle stages of trajectories. This indicates that it provides denser optimization signals around critical decision regions, leading to stable Pass@1 improvement. In contrast, random retry performs worse than full retry (MetaRL), suggesting that the gain comes from structured reflection.

Analysis of Training Interaction Cost. PivoARL reuses the correct prefix through pivotal retry, avoiding redundant interactions caused by full retries. To evaluate its interaction efficiency, we compare PivoARL with GRPO, GiGPO, and MetaRL, as shown in Figure 5. We normalize the number of training interactions by setting GRPO to 100%. Results show that PivoARL reduces interaction cost on average, achieving about 29.1% fewer interactions than GRPO and 13.5% fewer than MetaRL. This demonstrates that PivoARL improves performance while offering higher training efficiency.

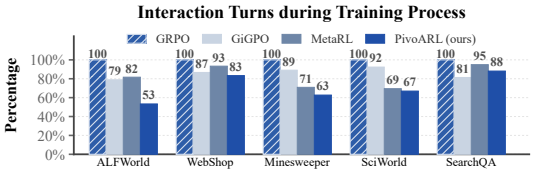
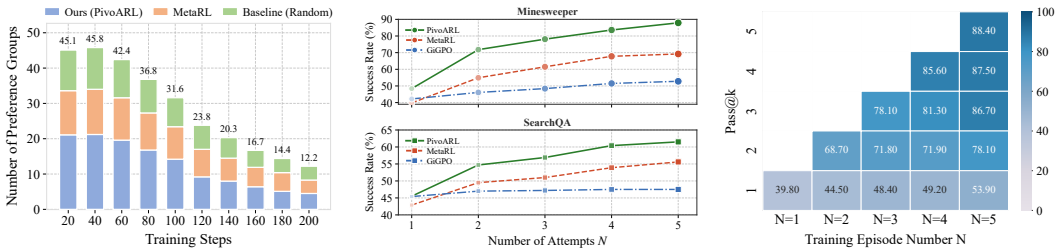


Figure 5: Comparison of interaction counts across training algorithms, with GRPO set to 100%.

Exhibits Stronger Test-time Scaling. To evaluate whether PivoARL can better utilize additional test-time attempts, we measure Pass@k by varying the number of rollouts during evaluation. As shown in Figure 7b, PivoARL consistently outperforms both standard RL and full-retry reflection baselines across all values of k. Averaged over Minesweeper and SearchQA, PivoARL improves from 47.0% Pass@1 to 74.7% Pass@5, while MetaRL improves from 41.4% to 62.4%. These results indicate that PivoARL exhibits stronger test-time scaling by learning to use additional attempts more effectively through reflective error localization.

Scaling Performance with Episode Nums. We further study the training-time scaling behavior of PivoARL by varying the number of training episodes (N). As shown in Figure 7c, PivoARL generally achieves stronger Pass@k performance as (N) increases. In particular, Pass@1 improves from 39.8% at (N=1) to 53.9% at (N=5), while the best overall result reaches 88.4% under the largest episode



(a) Change of Preference groups (b) Test-time Scaling (c) Train-time Scaling

Figure 7: Analysis of PivoARL mechanism: (a) preference-group dynamics, (b) test-time scaling, and (c) training-episode scaling on Minesweeper.

budget. These results suggest that increasing the number of training episodes provides richer retry and preference signals, enabling PivoARL to scale effectively during training.

PivoARL also improves pivotal-error localization accuracy to 71.5%, 68.5%, and 55.0% on Minesweeper, SearchQA, and ALFWorld, respectively (Appendix D.2), with further validation on multi-seed stability and cross-model generalization (Appendices D.4 and D.3).

5 Related Work

Agent Reinforcement Learning. RL plays a crucial role in helping LLM agents adapt to dynamic and open-ended environments He et al. (2025). Recent work such as RAGEN (Wang et al., 2025) integrates reasoning states with environment interactions into turn-level responses and trains with trajectory-level RL. To address sparse rewards in multi-turn interactions, IGPO (Wang et al., 2026a) introduces information-gain-based intrinsic rewards derived from the model's own confidence. To better capture multi-turn agent behaviors that trajectory-level RL struggles with, Tree-GRPO (Ji et al., 2025) employs tree-search-based group relative policy optimization to enable learning from intermediate decisions. In addition, GSPO (Zheng et al., 2025a) CISPO (MiniMax et al., 2025) and SAPO (Gao et al., 2025) focus on the stability of policy optimization, mitigating high-variance updates through different importance-ratio clipping mechanisms.

Self-Feedback Evolving Agents. Recently, a growing number of methods have begun to emphasize the role of interaction experience in the continual evolution of agents. *In-context learning* approaches such as Reflect-GRPO (Bensal et al., 2025) and RLTF (Song et al., 2026) treat natural language reflections as instructive experience (Shinn et al., 2023; Yang et al., 2026; Xiao et al., 2026). To internalize such short-term experience into the model's inherent capabilities, the *offline distillation* line of work—including ERL (Shi et al., 2026a), R^3L (Shi et al., 2026b), and Critique-GRPO (Zhang et al., 2025) uses corrected trajectories or critiques to guide offline optimization. Furthermore, SDAR (Lu et al., 2026) uses self-distillation to internalize in-context feedback into model parameters. MetaRL (Jiang et al., 2026) and MR-Search (Xiao et al., 2026) view a trajectory as a continuous learning process across multiple retry episodes, enabling cross-episode experience utilization.

6 Conclusion

We proposed PivoARL, a self-feedback reinforcement learning framework for exploiting failed trajectories in LLM agents. Instead of restarting from scratch, PivoARL locates the pivotal erroneous turn and retries from that state, thereby reusing correct prefixes and reducing redundant interactions. This design concentrates experience around critical errors and avoids misleading credit to erroneous suffixes. Experiments show that PivoARL improves performance while reducing interaction cost.

Limitation and Future Work. PivoARL relies on accurate pivotal-turn localization and introduces additional training cost. Localization errors may weaken local retry and credit assignment, especially in long-horizon tasks with ambiguous failures. Future work will explore more robust localization and real-world tool-use settings.

THE USE OF LARGE LANGUAGE MODELS

Throughout the preparation of this manuscript, large language models were used as auxiliary tools for language refinement, code development and debugging, and experimental assistance, including result inspection and analysis organization. The research questions, methodological design, experimental protocols, interpretation of results, and scientific conclusions were determined and verified by the authors. All AI-assisted code and experimental outputs were reviewed and validated by the authors, who take full responsibility for the content and results reported in this work.

REPRODUCIBILITY STATEMENT

The core methodology and training workflow of PivoARL are described in Section 3 and Appendix C. Detailed training hyperparameters and environment-specific configurations are provided in Appendix C.1 and Table 4, while benchmark and evaluation settings are described in Section 4.1. Complete interaction and reflection prompts are provided in Appendix F. Supplementary experiments, including pivotal-error localization, cross-model generalization, and multi-seed stability, are reported in Appendix D.4. We also provide anonymized source code in the Supplementary Material, together with executable training and evaluation scripts and a README for environment setup.

REFERENCES

Shelly Bensal, Umar Jamil, Christopher Bryant, Melisa Russak, Kiran Kamble, et al. Reflect, retry, reward: Self-improving LLMs via reinforcement learning. *arXiv preprint arXiv:2505.24726*, 2025.

Xin-Qiang Cai and Masashi Sugiyama. Vi-curl: Stabilizing verifier-independent rl reasoning via confidence-guided variance reduction. *arXiv preprint arXiv:2602.12579*, 2026.

Xin-Qiang Cai, Wei Wang, Feng Liu, Tongliang Liu, Gang Niu, and Masashi Sugiyama. Reinforcement learning with verifiable yet noisy rewards under imperfect verifiers. *arXiv preprint arXiv:2510.00915*, 2025.

Jiajun Chai, Guojun Yin, Zekun Xu, Chuhuai Yue, Yi Jia, Siyu Xia, Xiaohan Wang, Jiwen Jiang, Xiaoguang Li, Chengqi Dong, Hang He, and Wei Lin. Rlfactory: A plug-and-play reinforcement learning post-training framework for llm multi-turn tool-use, 2025. URL https://arxiv.org/abs/2509.06980.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In *Proceedings of 28th European Conference on Artificial Intelligence (EACL)*, 2025.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. Soft adaptive policy optimization. *arXiv preprint arXiv:2511.20347*, 2025.

Weiyang Guo, Jing Li, Wenya Wang, Yu Li, Daojing He, Jun Yu, and Min Zhang. Mtsa: Multi-turn safety alignment for llms through multi-round red-teaming. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26424–26442, 2025.

Weiyang Guo, Zesheng Shi, Liye Zhao, Jiayuan Ma, Zeen Zhu, Junxian He, Min Zhang, and Jing Li. E3-tir: Enhanced experience exploitation for tool-integrated reasoning. 2026.

Hang He, Chuhuai Yue, Chengqi Dong, Mingxue Tian, Hao Chen, Zhenfeng Liu, Jiajun Chai, Xiaohan Wang, Yufei Zhang, Qun Liao, et al. Localsearchbench: Benchmarking agentic search in real-world local life services. *arXiv preprint arXiv:2512.07436*, 2025.

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 International Conference on Computational Linguistics (COLING)*, pp. 6609–6625, 2020.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for llm agent reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

Yulun Jiang, Liangze Jiang, Damien Teney, Michael Moor, and Maria Brbić. Meta-RL induces exploration in language agents. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.

Bowen Jin, Hansi Zeng, Zhenrui Yue, 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.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 2019.

Guocong Li, Jinjian Zhang, Ping Wang, Dongnan Liu, Tian Liang, Qiuyi Qi, Hao Huang, Siyan Guo, Mutian Bao, Wei Zhou, Linjian Mo, Hongxia Xu, and Jian Wu. Mol: Adaptive mixture-of-length reasoning for efficient question answering with context. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=oWWAeLEdE3.

Yinghao Li, Haorui Wang, and Chao Zhang. Assessing logical puzzle solving in large language models: Insights from a minesweeper case study. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (ACL)*, 2024.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for LLM agents. *arXiv preprint arXiv:2601.02553*, 2026.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*, 2026.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.

MiniMax, :, Aili Chen, Aonian Li, Bangwei Gong, et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics (EMNLP)*, pp. 5687–5711, 2023.

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

Taiwei Shi, Sihao Chen, Bowen Jiang, Linxin Song, Longqi Yang, and Jieyu Zhao. Experiential reinforcement learning. *arXiv preprint arXiv:2602.13949*, 2026a.

Weijie Shi, Yanxi Chen, Zexi Li, Xuchen Pan, et al. R^3L: Reflect-then-retry reinforcement learning with language-guided exploration, pivotal credit, and positive amplification. *arXiv preprint arXiv:2601.03715*, 2026b.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. 2021.

Yuda Song, Lili Chen, Fahim Tajwar, Rémi Munos, Deepak Pathak, J. Andrew Bagnell, Aarti Singh, and Andrea Zanette. Expanding the capabilities of reinforcement learning via text feedback. *arXiv preprint arXiv:2602.02482*, 2026.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. Zerosearch: Incentivize the search capability of llms without searching. *arXiv preprint arXiv:2505.04588*, 2025.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics (TACL)*, 2022.

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, et al. Information gain-based policy optimization: A simple and effective approach for multi-turn search agents. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026a.

Rosie Wang et al. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

Ruoyao Wang, Peter A. Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 11279–11298, 2022.

Zihan Wang, Chi Gui, Xing Jin, Qineng Wang, Licheng Liu, Kangrui Wang, Shiqi Chen, Linjie Li, Zhengyuan Yang, Pingyue Zhang, Yiping Lu, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen-2: Reasoning collapse in agentic rl. *arXiv preprint arXiv:2604.06268*, 2026b.

Zihan Wang, Chi Gui, Xing Jin, Qineng Wang, Licheng Liu, Kangrui Wang, Shiqi Chen, Linjie Li, Zhengyuan Yang, Pingyue Zhang, Yiping Lu, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen-2: Reasoning collapse in agentic rl. *arXiv preprint arXiv:2604.06268*, 2026c.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

Teng Xiao, Yige Yuan, Hamish Ivison, Huaisheng Zhu, et al. Meta-reinforcement learning with self-reflection for agentic search. *arXiv preprint arXiv:2603.11327*, 2026.

Lu Yang, Zelai Xu, Minyang Xie, Jiaxuan Gao, et al. MAGE: Meta-reinforcement learning for language agents toward strategic exploration and exploitation. *arXiv preprint arXiv:2603.03680*, 2026.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

Zhiqi Yu, Zhangquan Chen, Mengting Liu, Heye Zhang, and Liangqiong Qu. Unveiling implicit advantage symmetry: Why grpo struggles with exploration and difficulty adaptation. *arXiv preprint arXiv:2602.05548*, 2026.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, et al. The landscape of agentic reinforcement learning for llms: A survey. *Trans. Mach. Learn. Res.*, 2026, 2026. URL https://openreview.net/forum?id=RY19y2RI1O.

Xiaoying Zhang, Yipeng Zhang, Hao Sun, Kaituo Feng, Chaochao Lu, Chao Yang, and Helen Meng. Critique-GRPO: Advancing LLM reasoning with natural language and numerical feedback. *arXiv preprint arXiv:2506.03106*, 2025.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025a.

Xuhui Zheng, Kang An, Ziliang Wang, Yuhang Wang, and Yichao Wu. Stepsearch: Igniting llms search ability via step-wise proximal policy optimization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025b.

APPENDIX

A Training Algorithm Pipeline **15**

B Baseline and Environments **16**

B.1 Experiment Environment 16

B.2 Introduction of Baseline 16

B.3 Comparison with Closely Related Retry-based RL Methods 17

C Details of Experiment Setup **18**

C.1 Training parameter config 18

C.2 Symbol Notation in this paper 18

C.3 Training Objective Details 19

D Supplementary Experiment **19**

D.1 Detailed Search Benchmark Result 19

D.2 Evaluation of Pivotal Error Localization 19

D.3 Cross-Model Generalization 20

D.4 Training Stability Across Random Seeds 21

E Theoretical Analysis **21**

E.1 Illustrative Analysis for Information Gain Analysis 21

E.2 Comparison of Information Gain with Other Methods 23

E.3 Connection to Step-Level Preference Optimization 23

F Prompt Used in Experiment **24**

F.1 Prompt in Minesweeper Task 24

F.2 Prompt in ALFWorld Task 24

F.3 Prompt in SearchQA Task 24

G Case Study **24**

G.1 Example in Minesweeper Environment 25

G.2 Example in ALFWorld Environment 25

G.3 Example in SearchQA Environment 25

A TRAINING ALGORITHM PIPELINE

We provide the complete training procedure of PivoARL in Algorithm 1. The pipeline consists of pivotal-aware self-feedback retry, pivotal-aware cross-episode credit assignment, and joint policy optimization with experience distillation. After each failed episode, the agent identifies the pivotal erroneous turn and retries from the corresponding state while reusing the correct prefix. By propagating useful retry signals to correct prefix decisions and isolating erroneous suffixes from future credit, PivoARL converts failed trajectories into targeted corrective signals for policy learning.

Algorithm 1 Training Workflow of PivoARL

Require: Initial policy π_θ; task set $\mathcal{Q}$; maximum retry budget N

Initialize retry buffer $\mathcal{D} \leftarrow \varnothing$
for each training iteration $i = 1, \ldots, K$ **do**
Phase 1: Self-Feedback Pivotal Retry
  for each task $x \in \mathcal{Q}$ **do**
    Sample initial rollout $\tau^{(0)} \sim \pi_\theta(\cdot \mid x)$ and evaluate $R(\tau^{(0)})$
    Initialize trajectory chain $\mathcal{T}_x \leftarrow \{\tau^{(0)}\}$
    for $n = 0, \ldots, N-2$ **do**
      if $\tau^{(n)}$ is successful **then**
        break
      end if
      Generate structured reflection

$$\Delta^{(n)}_{\text{reflect}} = \left(k^{*(n)}, \text{remark}^{(n)}\right) \sim \pi_\theta(\cdot \mid x, c^{(n)})$$

      Retry from pivotal state $s^{(n)}_{k^{*(n)}}$ by reusing prefix $\tau^{(n)}_{<k^{*(n)}}$
      Generate suffix conditioned on $\text{remark}^{(n)}$ to obtain $\tau^{(n+1)}$
      Evaluate $R(\tau^{(n+1)})$
      Append $\tau^{(n+1)}$ to $\mathcal{T}_x$
      Store $\left(\tau^{(n)}, \tau^{(n+1)}, k^{*(n)}, \text{remark}^{(n)}\right)$ in $\mathcal{D}$
    end for
  end for
Phase 2: Pivotal-Aware Credit Assignment
  for each trajectory chain $\mathcal{T}_x$ in $\mathcal{D}$ **do**
    Compute intra-episode returns $g^{(n)}_t$
    for each failed episode $\tau^{(n)}$ with pivot $k^{*(n)}$ **do**
      Compute pivotal-aware return

$$G^{(n)}_t = g^{(n)}_t + \mathbb{I}[t < k^{*(n)}]\,\gamma_{\text{traj}} G^{(n+1)}_{k^{*(n)}}$$

      Isolate suffix $t \geq k^{*(n)}$ from future retry credit
      Compute implicit reflection reward

$$r^{(n)}_{\text{reflect}} \leftarrow \mathbb{I}[\tau^{(n+1)} \text{ succeeds}] G^{(n+1)}_{k^{*(n)}}$$

    end for
    Estimate action advantages $\hat{A}^{(n)}_t$ from $G^{(n)}_t$
    Estimate reflection advantages $\hat{A}^{(n)}_{\text{reflect}}$ from $r^{(n)}_{\text{reflect}}$
  end for
Phase 3: Joint Policy Optimization
  Update π_θ using action and reflection advantages:

$$\mathcal{L}_{\text{total}} = -\mathbb{E}\left[\sum_{n,t} \log \pi_\theta\left(a^{(n)}_t \mid s^{(n)}_t\right) \hat{A}^{(n)}_t + \sum_n \log \pi_\theta\left(\Delta^{(n)}_{\text{reflect}} \mid x, c^{(n)}\right) \hat{A}^{(n)}_{\text{reflect}}\right]$$

end for
return Optimized policy π_θ

B BASELINE AND ENVIRONMENTS

B.1 EXPERIMENT ENVIRONMENT

SciWorld. ScienceWorld is a text-based virtual environment based on the standardized elementary science curriculum. Agents complete multi-step science tasks through textual observations and environment interactions.

MineSweeper. A partially-observable logic game where the goal is to reveal all safe cells without hitting hidden mines. Based on numerical clues indicating adjacent mines, the agent must use logical deduction to navigate the board. We utilize a 6×6 grid containing 3 mines.

Webshop (Yao et al., 2022). A text-based environment simulating an e-commerce interface. Given a natural language instruction with specific product attributes, the agent must search, filter, and navigate product pages to purchase the item that best matches the criteria. Success depends on how well the chosen product satisfies the original requirements.

ALFWorld (Shridhar et al., 2021). A partially-observable simulator for multi-step household tasks. Agents follow natural language instructions to navigate rooms and interact with objects (e.g., cleaning or heating items). We train on activities like "Pick," "Look," "Clean," and "Heat," then evaluate performance both in-distribution and out-of-distribution using "Cool" and "Pick2" tasks.

SearchQA. We evaluate our method on 7 open-domain QA tasks in a search-augmented setting. We use Wikipedia as the retrieval corpus and E5 as the dense retriever. For each query, the environment returns the top-3 passages, each truncated to 1024 characters. The agent can iteratively search, read evidence, and refine its answer, testing its multi-turn evidence integration ability.

Environment State Restoration. We implement environment-specific restoration interfaces for all agent benchmarks. For each trajectory, the environment state at historical interaction turns is cached or reconstructed through the benchmark-specific interface, enabling retry to resume from the state corresponding to the predicted pivotal turn. Therefore, the reused prefix serves to recover a consistent environment state rather than merely providing textual context, and only the suffix after the pivotal turn is regenerated.

B.2 INTRODUCTION OF BASELINE

GiGPO (Feng et al., 2025). Group-in-Group Policy Optimization (GiGPO) is a critic-free group-based RL method designed for long-horizon LLM agents. It estimates advantages at both the trajectory level and the step level by grouping actions that originate from repeated anchor states, thereby providing more fine-grained credit assignment under sparse or delayed rewards.

GSPO (Zheng et al., 2025a). Group Sequence Policy Optimization (GSPO) improves the stability of RL training for LLMs by replacing token-level importance ratios with sequence-level likelihood ratios. With sequence-level clipping and optimization, GSPO reduces high-variance updates and improves training efficiency, especially for large-scale and MoE-based models.

R3L (Shi et al., 2026b). R3L introduces Reflect-then-Retry to synthesize corrected trajectories from identified failure points, together with Pivotal Credit Assignment that masks the shared prefix and optimizes only the diverging suffix, and Positive Amplification that strengthens successful trajectories. Its original formulation uses trajectory-level advantages with turn-level masking and further incorporates auxiliary SFT losses to maintain reflection and retry abilities. Since its training framework differs from our step-level agent RL pipeline, we reproduce the core R3L algorithm within our framework rather than directly adopting its original implementation. In our reproduction, the auxiliary SFT objective introduces noticeable optimization instability when combined with our step-level RL updates, leading to weaker empirical performance. We evaluate both with and without this auxiliary objective and report the better-performing configuration in the main results.

Table 4: **Hyperparameters and configurations for different environments.** This table summarizes the specific settings used for training and evaluation across different task domains.

Config	SciWorld	MineSweeper	WebShop	ALFWorld	SearchQA
γ_{step}	0.95	0.95	0.95	0.95	0.95
γ_{traj}	0.6	0.6	0.6	0.6	0.6
optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
learning rate	1e-6	1e-6	1e-6	1e-6	1e-6
rollout n	8	8	8	8	5
total training steps	150	200	150	200	200
training batch size	16	16	8	8	16
PPO mini batch size	16	16	16	16	16
max prompt length	4096	4096	8192	4096	8192
max response length	1024	1024	1024	1024	1024
max turns	15	7	12	15	4

MetaRL (Jiang et al., 2026). Meta-RL, instantiated as LaMer, trains language agents to explore and adapt across episodes. It combines cross-episode reinforcement learning with in-context policy adaptation via reflection, enabling agents to learn from test-time feedback without gradient updates and improving exploration in long-horizon tasks.

Mem0+GRPO (Chhikara et al., 2025). Mem0 is a scalable long-term memory framework for LLM agents that extracts, stores, and updates salient information from past interactions. It maintains compact user- and task-specific memories through memory extraction, conflict , and relevance-based retrieval, enabling agents to reuse prior experience without directly appending full histories to the context. When paired with GRPO, Mem0 serves as the external memory module, while GRPO optimizes the policy using group-based reinforcement learning over memory-augmented trajectories.

SimpleMem+GRPO (Liu et al., 2026). SimpleMem is an efficient lifelong memory framework for LLM agents that compresses raw interaction histories into compact, structured memory units. It combines semantic structured compression, recursive memory consolidation, and adaptive query-aware retrieval to reduce redundant context while preserving task-relevant information. When paired with GRPO, SimpleMem serves as the memory module, while GRPO optimizes the policy using group-based reinforcement learning over retrieved memory-augmented trajectories.

SkillRL (Xia et al., 2026). SkillRL is a recursive skill-augmented reinforcement learning framework for LLM agents. Instead of storing raw trajectories as memory, it distills past experiences into a hierarchical skill library, retrieves task-relevant skills during interaction, and recursively evolves the skill bank together with the policy.

Table 5: **Comparison with closely related retry-based RL methods.** We compare retry scope, credit granularity, prefix handling, suffix treatment, and reflection optimization.

Characteristic (→) Method (↓)	Retry Scope	Credit Level	Prefix Handling	Suffix Treatment	Reflection Optimization
MetaRL	Full	Step	Re-generated	Joint Credit	Episode
R3L	Pivotal	Trajectory	Masked	Optimized	SFT
PivoARL (Ours)	**Pivotal**	**Step**	**Reused + Credited**	**Isolated**	**Implicit Return**

B.3 Comparison with Closely Related Retry-based RL Methods

MetaRL, R3L, and PivoARL all leverage retry trajectories to improve agent learning from failed interactions, but they differ in how retries are generated and how learning signals are assigned across episodes. MetaRL adopts a full-retry cross-episode paradigm, where each failed attempt is followed by a new episode from the initial state and later rewards are propagated across episodes.

Table 6: **Notation used in this paper.** Summary of key variables and operations in PivoARL. The *Definition* column indicates where each symbol first appears in the main text.

Symbol	Description	Definition
$x \sim \mathcal{D}$	Task sampled from the training task distribution	Sec. 2.1
π_θ	Trainable agent policy	Sec. 2.1
τ	Agent trajectory $\{(s_t, a_t, r_t)\}_{t=0}^{T-1}$	Sec. 2.1
$R(\tau)$	Episode-level return $\sum_{t=0}^{T-1} r_t$	Sec. 2.1
c	Interaction context composed of action-observation history	eq. (1)
remark	Self-feedback experience extracted from a failed attempt	eq. (1)
$\mathcal{T}$	Retry trajectory chain containing multiple episodes for one task	Sec. 3.1
$\tau^{(n)}$	The n-th episode in a retry trajectory chain	Sec. 3.1
$k^{*(n)}$	Pivotal erroneous turn in the n-th failed episode	eq. (4)
$\Delta^{(n)}_{\text{reflect}}$	Structured reflection containing $k^{*(n)}$ and $\text{remark}^{(n)}$	eq. (4)
$\tau^{(n+1)}$	Retry trajectory regenerated from the pivotal state	eq. (5)
$g_t^{(n)}$	Intra-episode discounted return from step t	eq. (6)
$G_t^{(n)}$	Pivotal-aware cross-episode return assigned to step t	eq. (6)–(7)
$\gamma_{\text{step}}, \gamma_{\text{traj}}$	Intra-episode and cross-episode discount factors	eq. (6)
$r^{(n)}_{\text{reflect}}$	Reflection reward derived from the subsequent retry return	eq. (8)
$\hat{A}_t^{(n)}$	Action advantage estimated from $G_t^{(n)}$	eq. (9)
$\hat{A}^{(n)}_{\text{reflect}}$	Reflection advantage estimated from $r^{(n)}_{\text{reflect}}$	eq. (9)
$IG(e, t)$	Per-turn information gain induced by experience guidance e	eq. (10)
$\rho(e)$	Effective signal density of experience guidance	Sec. 3.3

R3L uses the pivotal state primarily to determine where to regenerate, applying trajectory-level advantages to the diverging suffix while masking the shared prefix. In contrast, PivoARL additionally uses the pivotal state to determine how rewards propagate across episodes, crediting the reused prefix while isolating the failed suffix.

In contrast, PivoARL couples pivotal retry with fine-grained step-level credit assignment. It reuses the correct prefix, propagates useful future retry signals to prefix decisions, isolates erroneous suffixes from misleading cross-episode credit, and optimizes reflection through implicit returns rather than an auxiliary SFT objective. Table 5 summarizes these key differences.

C Details of Experiment Setup

C.1 Training parameter config

This subsection reports the training hyperparameters and configuration settings used for the four agent environments: SciWorld, MineSweeper, WebShop, and ALFWorld. To improve rollout efficiency in agentic loops, we use the non-thinking mode during trajectory generation. The experience guidance **remark** is attached to the original task prompt. For a fair comparison, we keep most optimization-related settings consistent across environments, while adapting environment-specific limits such as maximum turns and prompt length according to task complexity. In Table 4, we list the detailed training configuration parameters used in each environment.

C.2 Symbol Notation in this paper

We include the main notation for agent-environment interaction, self-feedback pivotal retry, pivotal-aware credit assignment, experience distillation, and information-gain analysis. The definitions Table 6 are intended to provide a compact reference for the method formulation and experimental discussion.

C.3 Training Objective Details

We follow GiGPO to compute the action-level advantage used in Eq. 9. Specifically, we retain the standard episode-level advantage computed from the trajectory reward, while applying our pivotal-aware return $G_t^{(n)}$ only to the step-level relative advantage.

First, the episode-level advantage is computed by normalizing the trajectory return within the retry group:

$$A^E(\tau^{(n)}) = \frac{R(\tau^{(n)}) - \text{mean}\left(\{R(\tau^{(m)})\}_{m=0}^{N-1}\right)}{F_{\text{norm}}\left(\{R(\tau^{(m)})\}_{m=0}^{N-1}\right)}, \tag{12}$$

where $R(\tau^{(n)})$ denotes the total return of trajectory $\tau^{(n)}$, and F_{norm} denotes the normalization factor, such as the standard deviation or a constant value.

For step-level credit assignment, we group actions generated from the same environment state:

$$\mathcal{G}^S(\tilde{s}) = \left\{ \left(a_t^{(n)}, G_t^{(n)} \right) \mid s_t^{(n)} = \tilde{s} \right\}. \tag{13}$$

Here, s_t denotes the underlying environment state rather than the full textual LLM context. Thus, reflection remarks do not affect state grouping, and the failed and retry-corrected actions at k^* are assigned to the same state group.

For each action $a_t^{(n)}$ in $\mathcal{G}_S(\tilde{s})$, the step-level advantage is computed using the pivotal-aware cross-episode return $G_t^{(n)}$:

$$A^S(a_t^{(n)}) = \frac{G_t^{(n)} - \text{mean}\left(\left\{ G_{t'}^{(m)} \mid \left(a_{t'}^{(m)}, G_{t'}^{(m)} \right) \in \mathcal{G}^S(\tilde{s}) \right\} \right)}{F_{\text{norm}}\left(\left\{ G_{t'}^{(m)} \mid \left(a_{t'}^{(m)}, G_{t'}^{(m)} \right) \in \mathcal{G}^S(\tilde{s}) \right\} \right)}. \tag{14}$$

Finally, the action-level advantage is obtained by combining the two terms:

$$\hat{A}_t^{(n)} = A_E(\tau^{(n)}) + \omega A_S(a_t^{(n)}). \tag{15}$$

where ω controls the weight of the step-level advantage. This advantage is then used as $A_t^{(n)}$ in the policy-gradient objective.

D Supplementary Experiment

D.1 Detailed Search Benchmark Result

We present a more detailed breakdown of the search benchmark results on Table 7, including the pass@1, pass@2, and pass@3 metrics.

D.2 Evaluation of Pivotal Error Localization

To directly evaluate the pivotal-error localization ability learned by different RL methods, we conduct an additional diagnostic experiment on MineSweeper and SearchQA. We first use the Qwen3-4B model to collect failed trajectories from the validation set, obtaining **100** trajectories for Minesweeper, ALFWorld and **200** trajectories for SearchQA. For each failed trajectory, we use DeepSeek-V4-Flash as a stronger external annotator. Given the complete trajectory and task outcome, the annotator is instructed to identify the earliest erroneous action that causes the subsequent trajectory to deviate from a successful solution. The annotated turn is treated as the reference pivotal index k^*_{ref}.

Evaluation Setup. We evaluate four policies with the same set of annotated failed trajectories: the original Qwen3-4B model and models trained with GRPO, MetaRL, and PivoARL. For each trajectory, all models receive the same interaction history and the same structured reflection instruction, and independently predict a pivotal turn $\hat{k}$. The base model represents the localization ability before

Table 7: Main results across various search benchmarks with Qwen3-4B. **Suc@1/2/3**: success rate in 1, 2 or 3 rollouts (%). The best results and second best results are highlighted in red and blue, respectively.

Methods	NQ†			TriviaQA⋆			PopQA⋆			HotpotQA†		
	Suc@1	Suc@2	Suc@3	Suc@1	Suc@2	Suc@3	Suc@1	Suc@2	Suc@3	Suc@1	Suc@2	Suc@3
Classic Open-Source Search Models												
Search-R1	46.3	48.3	48.3	61.6	63.2	63.6	47.4	48.8	48.9	46.7	47.2	47.3
ZeroSearch	48.8	51.2	51.2	61.6	62.8	63.1	47.4	48.2	48.3	45.9	47.5	48.1
StepSearch	48.8	49.3	49.3	64.3	65.4	65.8	47.9	47.9	47.9	45.1	48.1	48.8
RL-based Methods												
GRPO	48.2	49.2	49.7	62.0	63.4	63.7	49.8	50.0	50.0	46.9	47.3	47.3
GiGPO	48.7	49.3	49.7	65.5	66.9	67.2	47.9	48.8	49.1	47.2	48.2	48.2
GSPO	47.3	48.8	48.8	64.8	64.8	64.8	48.2	49.9	50.6	45.6	45.7	45.9
Reflect-Retry RL-based Methods												
MetaRL	44.8	50.2	51.7	62.7	66.9	67.8	44.7	49.2	49.8	45.4	52.7	54.6
PivoARL (Ours)	48.3	58.2	59.8	64.1	70.1	71.6	50.7	58.1	59.4	43.3	52.2	53.0

Methods	2Wiki⋆			MuSiQue⋆			Bamboogle			[—-]		
	Suc@1	Suc@2	Suc@3	Suc@1	Suc@2	Suc@3	Suc@1	Suc@2	Suc@3	Suc@1	Suc@2	Suc@3
Classic Open-Source Search Models												
Search-R1	47.6	51.2	52.4	22.9	22.9	22.9	44.0	46.4	46.4	–	–	–
ZeroSearch	42.7	45.8	47.3	23.0	23.1	23.1	41.6	43.2	46.4	–	–	–
StepSearch	44.6	47.0	50.0	20.0	22.2	22.4	48.0	51.2	51.2	–	–	–
RL-based Methods												
GRPO	42.9	45.0	46.4	19.3	20.7	20.7	46.4	47.2	47.2	–	–	–
GiGPO	43.6	47.8	47.9	19.3	20.0	20.0	45.6	48.0	48.0	–	–	–
GSPO	42.6	42.7	43.0	18.5	18.5	18.5	40.8	40.8	40.8	–	–	–
Reflect-Retry RL-based Methods												
MetaRL	41.2	52.5	55.2	17.0	23.7	24.4	44.8	51.2	53.6	–	–	–
PivoARL (Ours)	45.3	61.8	64.0	22.2	28.8	31.1	44.8	54.0	59.2	–	–	–

RL training, while GRPO, MetaRL, and PivoARL represent the corresponding abilities after RL optimization. We report exact pivotal localization accuracy:

$$\mathrm{Acc}_{\mathrm{pivot}} = \frac{1}{|\mathcal{D}_{\mathrm{fail}}|} \sum_{\tau \in \mathcal{D}_{\mathrm{fail}}} \mathbb{I}\left[\hat{k}_\tau = k^*_{\mathrm{ref},\tau}\right], \tag{16}$$

where $\mathcal{D}_{\mathrm{fail}}$ denotes the annotated failed-trajectory set.

Results. As shown in Fig. 8, standard RL training provides only limited improvements in pivotal-error localization, while PivoARL yields substantially higher accuracy across all three environments. On Minesweeper, the base model achieves 13.0% accuracy, while GRPO and MetaRL reach 22.0% and 24.5%, respectively; in contrast, PivoARL achieves 71.5%, improving over the base model and MetaRL by 58.5 and 47.0 percentage points. On SearchQA, the accuracies are 19.0%, 34.5%, and 22.5% for the base model, GRPO, and MetaRL, whereas PivoARL reaches 68.5%.

A similar trend is observed on ALFWorld, where PivoARL achieves 55.0%, compared with 11.0% for the base model, 27.0% for GRPO, and 34.0% for MetaRL. These results suggest that the gains of PivoARL are not merely due to generic RL optimization; instead, repeatedly identifying pivotal errors and learning from retry-dependent signals substantially strengthens the model's ability to localize critical failure points.

D.3 Cross-Model Generalization

To evaluate whether the gains of PivoARL are specific to Qwen3-4B, we conduct additional experiments with Qwen2.5-7B-Instruct across the four agent benchmarks. Following the main experimental setup, we compare GRPO, GiGPO, MetaRL, and PivoARL under the same backbone and training protocol.

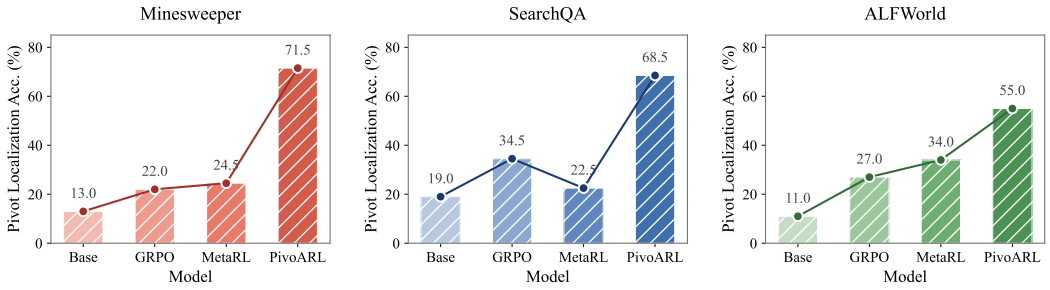


Figure 8: Pivotal-error localization accuracy before and after RL training. We evaluate the Qwen3-4B model and its GRPO-, MetaRL-, and PivoARL-trained variants on the same DeepSeek-annotated failed trajectories. (a) Minesweeper. (b) SearchQA. (C) ALFWorld.

Table 8: Cross-model results across four agent benchmarks with Qwen2.5-7B-Instruct. **Suc@1/2/3**: success rate in 1, 2 or 3 rollouts (%); **Avg.** denotes the mean of all success metrics across environments. The best results and second best results are highlighted in red and blue, respectively.

Methods	ALFWorld			WebShop			Minesweeper			SciWorld			Avg.
	Suc@1	Suc@2	Suc@3	Suc@1	Suc@2	Suc@3	Suc@1	Suc@2	Suc@3	Suc@1	Suc@2	Suc@3	
GRPO	81.4	83.7	84.8	64.7	65.4	65.9	38.3	41.8	42.1	77.4	80.4	82.8	67.4
GiGPO[NeurIPS 25]	86.8	89.7	91.7	**69.0**	72.8	73.4	43.1	47.6	48.8	84.4	85.7	86.7	73.3
MetaRL[ICLR 26]	84.8	89.1	92.0	67.3	72.3	74.7	42.4	49.8	51.7	82.3	89.6	93.2	74.1
PivoARL (Ours)	**87.4**	**90.5**	**93.0**	67.8	**76.3**	**81.2**	**45.6**	**54.4**	**61.7**	**87.4**	**92.0**	**95.9**	**77.8**

As shown in Table 8, PivoARL achieves the best performance on 11 out of 12 success metrics and improves the overall average from 74.1% (MetaRL) to 77.8%, corresponding to a 3.7 percentage-point gain. These results demonstrate that the effectiveness of pivotal retry generalizes beyond Qwen3-4B and is not tied to a specific backbone model.

D.4 Training Stability Across Random Seeds

To further examine the robustness of our results, we conduct three independent training runs with different random seeds on two representative settings, Minesweeper and SearchQA. We compare PivoARL with GiGPO and MetaRL under the same training configurations as the main experiments.

Table 9 reports the mean and standard deviation of Suc@1/2/3 across three runs. On Minesweeper, PivoARL achieves 48.7/72.0/77.8 on Suc@1/2/3, outperforming the strongest baseline by 5.6, 16.7, and 14.4 percentage points, respectively. On SearchQA, PivoARL obtains 45.9/54.1/57.3, with corresponding gains of 1.6, 5.0, and 5.8 percentage points over the best competing method. Overall, PivoARL consistently maintains its performance advantage across different random seeds, with relatively small standard deviations, indicating that the observed improvements are stable across independent training runs.

E Theoretical Analysis

E.1 Illustrative Analysis for Information Gain Analysis

Recall the per-turn information gain $\mathrm{IG}(e,t) = \log \pi_\theta(a_t^* \mid s_t, e) - \log \pi_\theta(a_t^* \mid s_t)$ and the effective signal density $\rho(e) = \frac{1}{T}|\{t : \mathrm{IG}(e,t) > \delta\}|$. We prove each case separately.

Case (i): Global reflection (MetaRL / LAMER). After a failed episode $\tau^{(n)}$, the agent generates a holistic reflection Δ summarizing the failure. This same Δ is prepended to the context at *every* turn $t \in \{0, \ldots, T-1\}$.

Table 9: Training stability across three random seeds on representative agent and search tasks. Results are reported as mean ± standard deviation. **Suc@1/2/3**: success rate in 1, 2, or 3 rollouts (%). The best results and second best results are highlighted in red and blue, respectively.

Methods	Minesweeper			SearchQA		
	Suc@1	Suc@2	Suc@3	Suc@1	Suc@2	Suc@3
GiGPO[NeurIPS 25]	43.1 ± 1.3	46.0 ± 0.7	48.6 ± 0.5	44.3 ± 0.5	47.4 ± 0.7	47.6 ± 0.9
MetaRL[ICLR 26]	38.7 ± 0.4	55.3 ± 0.6	63.4 ± 1.3	43.4 ± 1.0	49.1 ± 0.5	51.5 ± 0.7
PivoARL (Ours)	**48.7 ± 0.6**	**72.0 ± 1.1**	**77.8 ± 0.7**	**45.9 ± 1.4**	**54.1 ± 0.1**	**57.3 ± 0.6**

Under a bounded-influence assumption, the total positive influence of global guidance does not grow with the trajectory length:

$$\sum_{t=0}^{T-1} [\mathrm{IG}(\Delta, t)]_+ \leq C, \tag{17}$$

where $[x]_+ = \max(x, 0)$ and C is independent of T. For any fixed threshold $\delta > 0$,

$$\delta |\{t : \mathrm{IG}(\Delta, t) > \delta\}| \leq \sum_{t=0}^{T-1} [\mathrm{IG}(\Delta, t)]_+ \leq C. \tag{18}$$

Therefore,

$$\rho(\Delta) \leq \frac{C}{\delta T} = O(1/T). \tag{19}$$

Cross-trial noise accumulation. Consider the accumulated context at retry n:

$$\mathcal{H}^{(n)} = (\tau^{(0)}, \Delta^{(0)}, \ldots, \tau^{(n-1)}, \Delta^{(n-1)}). \tag{20}$$

Each failed trajectory $\tau^{(m)}$ contains both correct steps ($t < k^{*(m)}$) and erroneous steps ($t \geq k^{*(m)}$), but $\Delta^{(m)}$ does not distinguish between them. The noise tokens grow at least linearly:

$$|\text{noise tokens in } \mathcal{H}^{(n)}| \geq \sum_{m=0}^{n-1} (T^{(m)} - k^{*(m)}), \tag{21}$$

while the reflection signal is bounded by $|\Delta^{(m)}|$ per retry. The signal-to-noise ratio:

$$\mathrm{SNR}(\mathcal{H}^{(n)}) \leq \frac{n \cdot |\Delta|_{\max}}{\sum_m (T^{(m)} - k^{*(m)})}, \tag{22}$$

may remain low when the accumulated irrelevant context grows faster than the useful reflection signal, further weakening the effective guidance.

Case (ii): Experience retrieval (SkillsRL). At each turn t, the agent retrieves $e^* = \mathrm{Retrieve}(\mathcal{M}, s_t)$ from a memory bank $\mathcal{M} = \{(s_i, a_i, e_i)\}_{i=1}^{|\mathcal{M}|}$ via similarity $\mathrm{sim}(s_t, s_i)$. For planning tasks with combinatorially large state spaces (e.g., SciWorld with $|\mathcal{S}| = \Omega(\binom{100}{b} \cdot (100 - b))$), the match probability vanishes:

$$P(\exists\, e_i \in \mathcal{M} : \mathrm{sim}(s_t, s_i) > \delta) \leq \frac{|\mathcal{M}|}{|\mathcal{S}_\delta(s_t)|} \to 0, \tag{23}$$

where $\mathcal{S}_\delta(s_t)$ is the δ-neighborhood of s_t. When the retrieved experience does not match the current state:

$$\mathrm{sim}(s_t, s_{e^*}) \leq \delta \implies \pi_\theta(a_t^* \mid s_t, e^*) \approx \pi_\theta(a_t^* \mid s_t), \tag{24}$$

and therefore $\mathrm{IG}(e^*, t) \approx 0$. Since this holds for the vast majority of states encountered during rollout:

$$\mathbb{E}_{s_t \sim \pi_\theta}[\mathrm{IG}(e^*, t)] \approx 0. \tag{25}$$

This is the *template collapse* phenomenon (Wang et al., 2026b): the retrieved experience becomes a fixed template the model learns to ignore, contributing no actionable information to the decision process.

E.2 Comparison of Information Gain with Other Methods

We provide additional details for the empirical information-gain (IG) analysis in Fig. 3. We compare PivoARL with MetaRL and SkillsRL on the MINESWEEPER environment using their trained models. For each method, we compute IG on 500 trajectories from the validation set under the same evaluation protocol.

Reference action. For Eq. (10), the reference action a_t^* is obtained from a successful trajectory generated by the trained model on the validation set. We further apply an LLM-based filtering procedure to select high-quality successful trajectories as golden trajectories, and use the corresponding action at each turn as a_t^*. Given the experience guidance e, we compute

$$\mathrm{IG}(e, t) = \log \pi_\theta(a_t^* \mid s_t, e) - \log \pi_\theta(a_t^* \mid s_t).$$

Thus, IG measures how much the provided experience increases the likelihood of the reference action under the trained policy.

Aggregation and threshold. We pool the IG values from all turns of all 500 validation trajectories to obtain the empirical density distribution. We use a fixed empirical threshold $\delta = 0.15$ to distinguish low- and high-IG regions for analysis; it is used only for visualization and does not affect training. The reported peak and cumulative IG statistics are aggregated over the same validation trajectories.

E.3 Connection to Step-Level Preference Optimization

Pivotal retry can improve first-attempt performance because it naturally induces a step-level preference at the error boundary. Given a failed trajectory $\tau^{(n)}$, PivoARL identifies the pivotal turn $k^{*(n)}$ and retries from the same state. For brevity, we omit the episode superscript and write $k^* = k^{*(n)}$. Since the prefix before k^* is reused, the failed action and the retry-corrected action share the same state s_{k^*}:

$$a^- = a_{k^*}^{(n)}, \qquad a^+ = a_{k^*}^{(n+1)}.$$

If the retry succeeds while the original continuation fails, this induces a local preference

$$a^+ \succ a^- \quad \text{at} \quad s_{k^*}.$$

Proposition. Under a binary local outcome setting, the policy-gradient update induced by pivotal retry has the same preference direction as step-level DPO:

$$\nabla_\theta J_{\mathrm{pivot}} = w_{\mathrm{pivot}} \Big[\nabla_\theta \log \pi_\theta(a^+ \mid s_{k^*}, x) - \nabla_\theta \log \pi_\theta(a^- \mid s_{k^*}, x) \Big].$$

where $w_{\mathrm{pivot}} > 0$ is a scalar weight.

Proof. Consider the local binary outcome at the pivotal state:

$$G^+ = 1, \qquad G^- = 0,$$

where G^+ and G^- denote the returns of the corrected and failed continuations, respectively. Let

$$p^+ = \pi_\theta(a^+ \mid s_{k^*}, x), \qquad p^- = \pi_\theta(a^- \mid s_{k^*}, x).$$

Using the local expected return as the baseline, we have

$$b = p^+ G^+ + p^- G^- = p^+.$$

Thus, the corresponding local advantages are

$$A^+ = G^+ - b = p^-, \qquad A^- = G^- - b = -p^+.$$

The local policy-gradient update around s_{k^*} can be written as

$$\begin{aligned}\nabla_\theta J_{\text{pivot}} = {} & p^+ A^+ \nabla_\theta \log \pi_\theta(a^+ \mid s_{k^*}, x) \\ & + p^- A^- \nabla_\theta \log \pi_\theta(a^- \mid s_{k^*}, x).\end{aligned}$$

Substituting $A^+ = p^-$ and $A^- = -p^+$ gives

$$\begin{aligned}\nabla_\theta J_{\text{pivot}} &= p^+ p^- \nabla_\theta \log \pi_\theta(a^+ \mid s_{k^*}, x) \\ &\quad - p^- p^+ \nabla_\theta \log \pi_\theta(a^- \mid s_{k^*}, x) \\ &= p^+ p^- \Big[\nabla_\theta \log \pi_\theta(a^+ \mid s_{k^*}, x) \\ &\quad - \nabla_\theta \log \pi_\theta(a^- \mid s_{k^*}, x)\Big].\end{aligned}$$

Therefore, $w_{\text{pivot}} = p^+ p^- > 0$, and the proposition holds.

This gradient has the same direction as the step-level DPO gradient. For the same preference pair (a^+, a^-), step-level DPO yields

$$\begin{aligned}\nabla_\theta J_{\text{DPO}} = w_{\text{DPO}} \Big[& \nabla_\theta \log \pi_\theta(a^+ \mid s_{k^*}, x) \\ & - \nabla_\theta \log \pi_\theta(a^- \mid s_{k^*}, x)\Big].\end{aligned}$$

where $w_{\text{DPO}} > 0$ depends on the preference margin. Thus, pivotal retry induces a gradient direction consistent with step-level preference optimization under this simplified local setting.

Pivotal credit isolation prevents positive retry returns from leaking to the erroneous suffix containing a^-. Thus, PivoARL preserves a clean preference signal at k^*, encouraging the policy to choose a^+ in the first attempt and improving Succ@1.

F Prompt Used in Experiment

This section presents the prompts used in our experiments. For each environment, we provide a play prompt for agent-environment interaction and a reflect prompt for pivotal error localization.

F.1 Prompt in Minesweeper Task

Figure 9 shows the play prompt used for Minesweeper. Figure 10 shows the pivotal reflect prompt used for Minesweeper.

F.2 Prompt in ALFWorld Task

Figure 11 shows the play prompt used for ALFWorld. Figure 12 shows the pivotal reflect prompt used for ALFWorld.

F.3 Prompt in SearchQA Task

Figure 13 shows the play prompt used for SearchQA. Figure 14 shows the pivotal reflect prompt used for SearchQA.

G Case Study

This section provides qualitative examples of PivoARL across different environments. Each case illustrates how the agent identifies the pivotal failure point and improves the subsequent retry.

G.1 Example in Minesweeper Environment

Figure 15 shows a representative case study in the Minesweeper environment.

G.2 Example in ALFWorld Environment

Figure 16 shows a representative case study in the ALFWorld environment.

G.3 Example in SearchQA Environment

Figure 17 shows a representative case study in the SearchQA environment.

Minesweeper Play Prompt

```
You are an expert agent operating in the Minesweeper game.
You will be given a two dimensional {board_size} by
{board_size} board, with {n_mines} hidden mines.
The rows and columns are indexed from 1 to {board_size}.
# Cell States
• Unopened cells (?): cells that are yet to be revealed and
  may contain a mine.
• Blank cells (.): opened and non-mine cells, and they have
  no neighboring mines.
• Numbered cells (1-8): opened and non-mine cells, and
  the number indicates how many mines are in the eight
  neighboring cells.
# Your Goal
• Clear the board by revealing all the cells that don't
  contain mines.
• Use clues about the number of neighboring mines to reason.
# Reveal Rules
• Blank cell (.): Triggers an auto-cascade of contiguous
  blank cells.
• Numbered cell (1{8): Only that single cell is revealed.
• Mine (*): The game ends immediately in a loss.
# Observation The initial state of the game is:
{init_observation}{past_trajectories}{current_trajectory}
Now it's your turn to make a move.
```

Figure 9: The system prompt template for the Minesweeper agent.

Minesweeper Pivotal Reflection Prompt

```
Now reflect on the past experience:
• First, analyze each turn's decision and identify the root
  cause of failure.
• Then determine the earliest action number where the
  trajectory went wrong
• (the first shown action that should be changed).
• Finally, provide your reflection with the retry point.
Your response MUST end with:
1. A <turn_idx> tag containing the earliest turn number to
   retry from.
2. A <remark> tag containing your concise reflection and
   improved plan.
Format:
<turn_idx>integer</turn_idx>
<remark>your concise reflection and improved plan </remark>
Where:
• The text inside <remark> is a concise improved plan that
  accounts for the identified mistake.
• For example, if Action 3 is the first wrong action, output
  <turn_idx>3</turn_idx>.
• The integer inside <turn_idx> must be from 1 to {max_turn},
  matching the action numbers shown above.
```

Figure 10: The pivotal reflection prompt template for the Minesweeper agent.

ALFRED Play Prompt

```
You are an expert agent operating in the ALFRED Embodied
Environment.
{init_observation}{past_trajectories_reflections}{trajectory}
```

 # Admissible Actions

```
Your admissible actions of the current situation are:
[{admissible_actions}]
```

 # Your Task

```
Now it's your turn to take an action.

• If previous reflections are provided, use them as guidance
  but re-evaluate the current state and justify your choice
  with your own reasoning.
• Do not copy a prior reflection or past action sequence
  verbatim; adapt principles to the current context.
• Your response should first provide step-by-step reasoning
  about the current situation.
• Once you have finished your reasoning, choose an admissible
  action for the current step and present it within <action>
  </action> tags.
```

Format:

```
your step-by-step reasoning
<action>admissible action</action>
```

Figure 11: The play prompt template for the ALFRED embodied agent.

ALFWorld Pivotal Reflection Prompt

Now reflect on the past experience:
- First, analyze each turn's decision and identify the **root cause** of failure.
- Then determine the **earliest action number** where the trajectory went wrong
- (the first shown action that should be changed).
- Finally, provide your reflection with the retry point.

Your response MUST end with:
1. A <turn_idx> tag containing the earliest turn number to retry from.
2. A <remark> tag containing your concise reflection and improved plan.

Format:
<turn_idx>integer</turn_idx>
<remark>your concise reflection and improved plan </remark>

Where:
- The text inside <remark> is a concise improved plan that accounts for the identified mistake.
- For example, if Action 3 is the first wrong action, output <turn_idx>3</turn_idx>.
- The integer inside <turn_idx> must be from 1 to {max_turn}, matching the action numbers shown above.

Figure 12: The pivotal reflection prompt template for the ALFRED embodied agent.

SearchQA Play Prompt

You are an expert agent tasked with answering the given question step-by-step.
Your question: {task_description}

 # Your Task
Now it's your turn to respond for the current step.
You should first conduct reasoning process. This process MUST be enclosed within <think> </think> tags.

 # Action Space
After completing your reasoning, choose only one of the following actions. Do not perform both:
1. If you lack some knowledge, call a search engine to get external information using the format: <search> your query </search>.
2. If you have enough knowledge to answer confidently, provide your final answer within <answer> </answer> tags, without detailed illustrations.

 # Reflection
On trial #{traj_idx}, the task was not solved well enough.
Your reflection is:
{reflection}

Format:
<think>your reasoning process</think>
<search>your query</search> or <answer>final answer</answer>

Figure 13: The play prompt template for the SearchQA agent.

SearchQA Pivotal Reflection Prompt

```
You are an expert agent tasked with answering the given
question step-by-step.
Your question: {task_description}
# Your Task
You will be given the history of a past attempt.
Your job is to:
1. Reflect on the past search process and identify the
   earliest turn where a wrong or inefficient decision was
   made.
2. Determine the earliest turn number whose action should be
   changed.
3. Devise a concise improved plan starting from that turn.
# Past Experience
{current_trajectory}
The task is NOT successfully completed.
Now reflect on the past experience:
• First, analyze the search process turn by turn and identify
  the root cause of failure.
• Then determine the earliest action number where the
  trajectory went wrong.
• The earliest action is the first shown search or answer
  step that should be changed.
• Focus on the earliest pivotal mistake, not a later
  downstream consequence.
• Finally, provide a concise improved plan for retrying from
  that turn.
Format:
<turn_idx>integer</turn_idx>
<remark>
your concise reflection and improved plan
</remark>
Where:
• The text inside <remark> should summarize the corrected
  search strategy, missing checks, or improved answer
  criterion for the retry.
• The integer inside <turn_idx> must be from 1 to {max_turn}.
• Use 1 if the very first shown step should be changed.
• For example, if Step 3 is the first wrong search or answer
  decision, output <turn_idx>3</turn_idx>.
```

Figure 14: The pivotal reflect prompt template for the SearchQA agent.

Case: Minesweeper (Pivotal Retry)

Task: Reveal safe cells and avoid hidden mines (6×6 board, 3 mines).

Failed Agent Trajectory in First Attempt

Step 1 **Obs:**

```
Row 1:  2 ?  ?  ?  ?  ?
Row 2:  ?  ?  ?  ?  ?  ?
Row 3:  ?  1 ?  ?  ?  ?
Row 4:  ?  ?  ?  ?  ?  ?
Row 5:  ?  ?  ?  ?  ?  ?
Row 6:  ?  ?  ?  ?  ?  ?
```

Think: Step-by-Step Reasoning: We are given a 6x6 Minesweeper board with 3 hidden mines. The only known cell is at (3,2), which is a numbered cell with the value '1' ...
Action: `(3, 3)`

Step 2 **Obs:**

```
Row 1:  .  .  1 ?  1 .
Row 2:  .  .  1 1 1 .
Row 3:  1 1 .  .  .  .
Row 4:  ?  1 .  .  .  .
Row 5:  ?  2 .  .  .  .
Row 6:  ?  1 .  .  .  .
```

Think: Step-by-Step Reasoning: Let's analyze the current state of the board after the first action. We need to deduce where the remaining **3 ...
Action: `(3, 2)`

Step 3 **Obs:**

```
Row 1:  .  .  1 ?  1 .
Row 2:  .  .  1 1 1 .
Row 3:  1 1 .  .  .  .
Row 4:  ?  1 .  .  .  .
Row 5:  ?  2 .  .  .  .
Row 6:  ?  1 .  .  .  .
```

Think: Step-by-Step Reasoning: Let's analyze the current state of the board after the first action. We need to deduce where the remaining **3 ...
Action: `(4, 1)`

Failed First attempt fails after an unsafe pivot decision branch.

Self-Reflect on Failed Trajectory

Reflection **Reasoning:** After Action 1 clicking (3,3), the board reveals substantial information. Examining the neighbors of (3,1)=1: among its 8 surrounding cells, (2,1), (2,2), and (3,2) are all revealed ...
Retry Turn: `<turn_idx>3</turn_idx>`
Experience: The mistake was revealing (4, 1) which turned out to be a mine. The improved plan is to avoid guessing based on insufficient information and instead look for cells with unique numbers that can be deduced safely.

Success Agent Trajectory in Reflect Retry

Step 3 **Obs:**

```
Row 1:  .  .  1 ?  1 .
Row 2:  .  .  1 1 1 .
Row 3:  1 1 .  .  .  .
Row 4:  ?  1 .  .  .  .
Row 5:  ?  2 .  .  .  .
Row 6:  ?  1 .  .  .  .
```

Think: ¡reflection¿The previous attempt revealed a mine at (4, 1), which was a mistake. This highlights the need to be cautious and rely on logical deduction rather than random guessing.Looking at the current board state, I notice that some cells have been revealed and are numbered.
Action: `(5, 1)`

Success Reflect retry succeeds from the same pivot observation state.

Figure 15: Case: Minesweeper (Pivotal Retry)

Case: ALFWorld SprayBottle (Pivotal Retry)

Task: Put two spraybottles in a cabinet.

Failed Agent Trajectory in First Attempt

Step 1	**Obs:** Room initialized; countertop 1 contains spraybottle 1 and spraybottle 2. **Think:** I need to locate and collect two spraybottles before placing them in a cabinet. **Action:** `go to countertop 1`
Step 2	**Obs:** You arrive at countertop 1. Spraybottle 1 and spraybottle 2 are available. **Think:** I will first pick up spraybottle 1. **Action:** `take spraybottle 1 from countertop 1`
Step 3	**Obs:** You pick up spraybottle 1 from countertop 1. **Think:** I now hold spraybottle 1 and should move to a cabinet. **Action:** `go to cabinet 1`
Step 4	**Obs:** You arrive at cabinet 1. The cabinet is open and empty. **Think:** I currently hold spraybottle 1 and need to place it inside the cabinet. **Action:** `open cabinet 1`
Step 5	**Obs:** *Nothing happens.* **Think:** I am still holding spraybottle 1, but the previous action did not make progress. **Action:** `take spraybottle 1 from countertop 1`
Result: Failed	The first attempt fails after the pivotal mistake at Step 4 and its subsequent erroneous continuation.

Self-Reflect on Failed Trajectory

Reflection	**Reasoning:** The earliest mistake occurs at Step 4. At this state, cabinet 1 is already open and I am holding spraybottle 1, but I redundantly attempt to open the cabinet again instead of placing the spraybottle inside. **Retry Turn:** Step 4 **Experience:** When the cabinet is already open and the target object is held, place the object directly into the cabinet before fetching the second spraybottle.

Successful Agent Trajectory after Pivotal Retry

Step 4	**Obs:** You arrive at cabinet 1. The cabinet is open and empty. **Think:** I am holding spraybottle 1, so I should place it directly into the open cabinet. **Action:** `put spraybottle 1 in/on cabinet 1`
Step 5	**Obs:** You put spraybottle 1 in/on cabinet 1. **Think:** Spraybottle 1 is now in the cabinet. I need to fetch spraybottle 2. **Action:** `go to countertop 1`
Step 6	**Obs:** You arrive at countertop 1. Spraybottle 2 is available. **Think:** I need to pick up the remaining spraybottle. **Action:** `take spraybottle 2 from countertop 1`
Step 7	**Obs:** You pick up spraybottle 2 from countertop 1. **Think:** I now hold spraybottle 2 and should return to cabinet 1. **Action:** `go to cabinet 1`
Step 8	**Obs:** You arrive at cabinet 1. The cabinet is open and contains spraybottle 1. **Think:** I am holding spraybottle 2 and can now complete the task. **Action:** `put spraybottle 2 in/on cabinet 1`
Result: Success	The retry succeeds by restarting from the same pivotal environment state at Step 4.

Figure 16: Case study of pivotal retry in ALFWorld. The agent identifies Step 4 as the earliest erroneous action and retries from the same environment state, reusing the correct prefix while replacing the erroneous continuation.

Case: Complex Search QA (Pivotal Retry)

Task: How tall is the ride that is nearly identical to Goliath at Six Flags Magic Mountain?

Failed Agent Trajectory in First Attempt

Step 1	**Obs:** Search results describe Goliath at Six Flags Magic Mountain as a steel hypercoaster and state that it is "nearly identical to Titan at Six Flags Over Texas." **Think:** The target ride is therefore Titan at Six Flags Over Texas. I should verify this relationship before searching for its height. **Action:** `search: Titan Six Flags Over Texas nearly identical to Goliath`
Step 2	**Obs:** The returned result confirms that Titan at Six Flags Over Texas and Goliath at Six Flags Magic Mountain are nearly identical hypercoasters. **Think:** The relationship is confirmed, but I will search for Goliath's height instead of Titan's height. **Action:** `search: How tall is Goliath at Six Flags Magic Mountain?`
Step 3	**Obs:** The search result describes Goliath's features but does not provide a reliable numeric height in the returned snippet. **Think:** Since the height is not explicitly given, I will rely on prior knowledge. **Action:** `answer: 175 feet`
Result: Failed	The agent searches for the wrong ride after correctly identifying Titan as the target, and then produces an unverified answer.

Self-Reflection on Failed Trajectory

Reflection	**Reasoning:** Step 1 correctly identified Titan at Six Flags Over Texas as the ride nearly identical to Goliath at Six Flags Magic Mountain. The pivotal mistake occurs at Step 2, where the agent searches for Goliath's height instead of Titan's. This leads to an irrelevant result and an unverified final answer. **Pivot Turn:** Step 2 **Improved Plan:** Reuse the verified relationship from Step 1 and replace Step 2 with a direct search for the height of Titan at Six Flags Over Texas.

Successful Pivotal Retry from Step 2

Prefix	**Reused:** Step 1 is retained because it correctly identifies Titan at Six Flags Over Texas as the target ride.
Step 2 (corrected)	**Think:** Since Titan is the target ride, I should directly search for Titan's height. **Action:** `search: height of Titan at Six Flags Over Texas`
Step 3	**Obs:** The search result states that Titan at Six Flags Over Texas stands 245 feet tall and has a 255-foot drop. **Think:** Titan is confirmed to be 245 feet tall, which directly answers the question. **Action:** `answer: 245 feet`
Result: Success	Pivotal retry preserves the correct Step 1 prefix and regenerates only the erroneous suffix from Step 2.

Figure 17: SearchQA case study of pivotal retry. PivoARL preserves the correct search prefix, identifies the earliest erroneous search decision, and regenerates only the suffix from the pivotal turn.