RETROAGENT: FROM SOLVING TO EVOLVING VIA RETROSPECTIVE DUAL INTRINSIC FEEDBACK

Anonymous authors
Paper under double-blind review

ABSTRACT

Standard reinforcement learning (RL) for large language model (LLM) agents primarily optimizes extrinsic task rewards, often favoring one-off task completion over continual adaptation. This paradigm can cause premature convergence to suboptimal policies and leaves useful experience only implicitly encoded in model parameters, limiting its retrieval and reuse for future decisions. We introduce RETROAGENT, an online RL framework that trains agents to master interactive environments ***not merely by solving tasks, but by evolving across episodes***. Inspired by human retrospective self-improvement, RETROAGENT augments extrinsic rewards with hindsight-generated dual intrinsic feedback: (i) *Intrinsic Numerical Feedback*, which rewards beneficial exploration by measuring incremental subtask progress relative to prior attempts; and (ii) *Intrinsic Language Feedback*, which distills successes and failures into reusable textual lessons for explicit experience reuse. To leverage these lessons effectively, we propose *Similarity & Utility-Aware Upper Confidence Bound* (SIMUTIL-UCB), a retrieval strategy that balances semantic relevance, historical utility, and exploration. Across four challenging agentic benchmarks, RETROAGENT achieves new state-of-the-art (SOTA) performance, outperforming GRPO by $+18.3\%$ on ALFWorld, $+15.4\%$ on WebShop, $+27.1\%$ on Sokoban, and $+8.9\%$ on MineSweeper, while demonstrating strong test-time adaptation and out-of-distribution (OOD) generalization.

1 INTRODUCTION

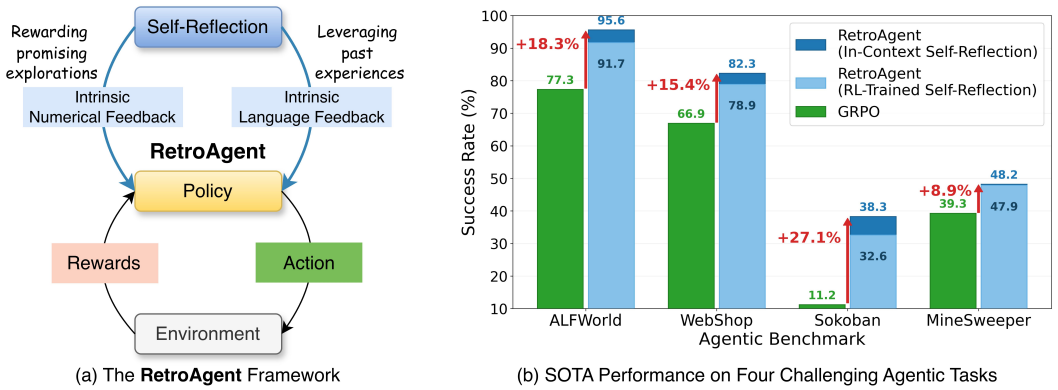
Reinforcement learning (RL) (Sutton et al., 1998) is a central paradigm for training large language model (LLM) agents through interaction with complex environments (Ouyang et al., 2022; Comanici et al., 2025; Singh et al., 2025; Liu et al., 2025b). However, optimizing extrinsic task-success rewards alone provides limited guidance for learning from unsuccessful attempts or explicitly reusing experience across interactions (Abel et al., 2023; Weng, 2026; Rafiee & Sutton, 2026). This creates two challenges. First, sparse rewards may fail to distinguish productive exploration from repetitive failure, encouraging over-exploitation of early successes and premature convergence (Kirk et al., 2024). Second, experience retained only in model parameters is difficult to retrieve, inspect, and explicitly reuse for subsequent policy improvement (Lin, 1992; Graves et al., 2014).

Prior work addresses these challenges along two largely separate directions. Exploration-oriented methods encourage broader search through cross-episode meta-RL (Jiang et al., 2025; Beck et al., 2025) or uncertainty-modulated rewards (Wang et al., 2025b), but do not necessarily preserve explicit, reusable lessons. Memory-augmented methods instead store trajectories (Goyal et al., 2022; Wu et al., 2025; Liu et al., 2026b) or distilled skills, rules, and lessons (Anthropic, 2025; Wang et al., 2025c; Liu et al., 2026b; Xia et al., 2026). However, when retrieval relies on fixed similarity metrics without accounting for downstream utility or policy updates, memory serves primarily as *passive context augmentation*. Noisy retrieval and excessive reliance on past experience can then undermine decision-making and discourage further exploration.

These limitations motivate a central question: *Can an agent combine active exploration with utility-aware experience reuse to continually improve under sparse external rewards, without manually engineered verifiers?*

To answer this question, we introduce RETROAGENT (Figure 1), an online RL framework inspired by human retrospective reflection (Lyons & Zelazo, 2011; Liu & van der Schaar, 2025). RETROAGENT

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.



(a) The **RetroAgent** Framework (b) SOTA Performance on Four Challenging Agentic Tasks

Figure 1: RETROAGENT overview and performance. (a) RETROAGENT combines online RL with retrospective reflection to produce *dual intrinsic feedback*: an adaptive exploration bonus without manual dense rewards, and reusable lessons in explicit memory with utility-aware retrieval. (b) Built on Qwen-2.5-7B-Instruct, RETROAGENT outperforms standard GRPO and achieves SOTA results across four agentic benchmarks.

trains agents ***not merely to solve tasks, but to evolve*** [1] ***across episodes*** by combining extrinsic task-success rewards with two complementary forms of retrospective intrinsic feedback. After each episode, the agent analyzes its trajectory in light of previous attempts to assess exploratory progress and extract reusable lessons, without predefined subtask verifiers or dense reward annotations. (i) **Intrinsic Numerical Feedback** assigns scalar rewards to productive exploration that sparse task-success rewards may fail to distinguish from unproductive failures. Because capability improvement is difficult to measure directly, RETROAGENT uses retrospectively assessed progress relative to prior attempts as a proxy, crediting attempts that overcome earlier bottlenecks or explore new task-relevant interactions. For example, locating a previously unfound target item may constitute progress even if the purchase ultimately fails. This comparative signal encourages exploration without assuming a predefined sequence of subtasks. (ii) **Intrinsic Language Feedback** distills actionable procedural lessons from successes and informative failures into an explicit memory buffer. To retrieve useful lessons, we propose *Similarity & Utility-Aware Upper Confidence Bound* (SIMUTIL-UCB), which combines semantic relevance with empirically estimated utility. Following UCB principles (Auer et al., 2002), it balances exploiting high-utility lessons with exploring under-tested ones.

We study two RETROAGENT variants: *in-context* self-reflection and *RL-trained* self-reflection jointly optimized with the decision policy. Although compatible with multiple RL algorithms, our implementation uses GRPO (Shao et al., 2024b) for decision-making and REINFORCE (Williams, 1992) for reflection. We evaluate Qwen-2.5-7B-Instruct (Qwen et al., 2025) and Llama-3.1-8B-Instruct (Grattafiori et al., 2024) on ALFWorld (Shridhar et al., 2021), WebShop (Yao et al., 2022a), Sokoban (Racanière et al., 2017), and MineSweeper (Li et al., 2024). Across all four benchmarks, RETROAGENT outperforms RL fine-tuning, memory-augmented RL, exploration-guided RL, and meta-RL baselines, improving prior SOTA success rates by approximately 10% on WebShop and 16% on Sokoban, with strong test-time adaptation and OOD generalization.

Our contributions are fourfold: (i) RETROAGENT, an online RL framework for cross-episode policy improvement through retrospective self-reflection; (ii) dual intrinsic feedback that rewards productive exploration and extracts reusable lessons under sparse external rewards; (iii) SIMUTIL-UCB, a retrieval strategy balancing semantic relevance, empirical utility, and exploration of under-tested lessons; and (iv) empirical gains across four agentic benchmarks, with SOTA results in the evaluated in-distribution and OOD settings.

2 RELATED WORK

Reinforcement Learning for LLM Agents. RL increasingly supports agent training in multi-turn, dynamic environments (Wang et al., 2025d; Putta et al., 2025; Liu et al., 2025a;b). ArCHer (Zhou

[1] We use *evolution* to denote task solving coupled with ongoing adaptation (Hinton et al., 1987).

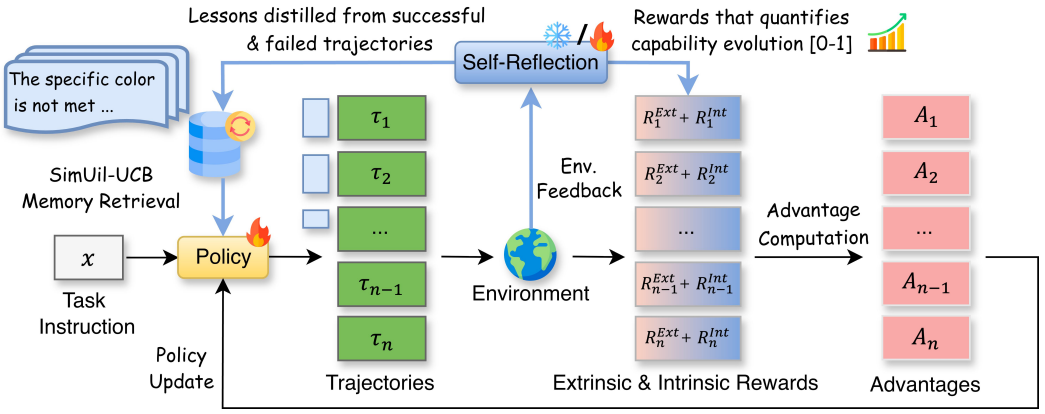


Figure 2: Overview of RETROAGENT. After each episode, self-reflection analyzes the trajectory in light of prior attempts to produce complementary intrinsic feedback: (i) *Numerical Feedback* rewards promising partial progress to guide exploration under sparse task-success rewards; and (ii) *Language Feedback* distills reusable lessons from successes and failures into explicit memory. The proposed SIMUTIL-UCB retrieves lessons by balancing semantic relevance, empirical utility, and exploration of under-tested lessons.

et al., 2024) uses hierarchical value functions for WebShop (Yao et al., 2022b), while LOOP (Chen et al., 2025) combines PPO (Schulman et al., 2017) with leave-one-out advantage estimation for AppWorld (Trivedi et al., 2024). Building on GRPO (Shao et al., 2024a), GiGPO (Feng et al., 2025) introduces two-level advantage estimation; related work explores turn-level reward shaping (Wei et al., 2025) and stepwise progress attribution (Wang et al., 2025a). Meta-RL (Beck et al., 2025) offers a complementary approach: LAMER (Jiang et al., 2025) trains across episodes to support active test-time exploration. These methods primarily rely on environmental feedback, motivating intrinsic signals beyond sparse task rewards (Liu & van der Schaar, 2025). Alongside work on intrinsic motivation (Gao et al., 2025) and entropy-modulated policies (Wang et al., 2025b), RETROAGENT contributes retrospective self-reflection that jointly produces exploration rewards and reusable lessons for cross-episode adaptation. See Appendix L for further discussion.

3 RETROAGENT

3.1 GENERAL OVERVIEW

Problem Formulation. We model the LLM agent's multi-turn interaction with its environment as a Markov Decision Process (MDP) (Sutton et al., 1998), defined by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $P(s_{t+1} \mid s_t, a_t)$ the environment's transition dynamics, $R(s_t, a_t)$ the reward function, and $\gamma \in [0, 1]$ the discount factor. At each step $t = 0, \ldots, T-1$, the agent observes state $s_t \in \mathcal{S}$ and samples action $a_t \in \mathcal{A}$ from its policy $\pi_\theta(\cdot \mid s_t)$. In the LLM agent setting, the state is the concatenation of all preceding observations and actions: $s_t = (o_0, a_0, \ldots, a_{t-1}, o_t)$. Executing a_t yields reward $r_{t+1} = R(s_t, a_t)$ and successor state $s_{t+1} \sim P(\cdot \mid s_t, a_t)$, producing a trajectory $\tau = (s_0, a_0, r_1, \ldots, s_{T-1}, a_{T-1}, r_T)$. With purely extrinsic rewards $r_{t+1} = r^{\text{ext}}_{t+1}$, the standard objective is to maximize the expected discounted return: $\mathcal{J}_{\text{Standard}}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\cdot|x) \times P}[\, G_0 \,] = \mathbb{E}_{\tau \sim \pi_\theta(\cdot|x) \times P}\left[\sum_{t=0}^{T-1} \gamma^t \, r^{\text{ext}}_{t+1} \right]$, where $x = o_0$ is the task instruction drawn from the training set $\mathcal{D}$, and $\tau \sim \pi_\theta(\cdot \mid x) \times P$ denotes that trajectories are generated jointly by the policy and the environment dynamics. In practice, extrinsic rewards are sparse: a non-zero terminal reward R^{ext} is provided only when the episode ends, either upon successful task completion or upon exceeding the allowed number of steps. To simplify credit assignment, we broadcast this terminal reward uniformly across all steps, setting $r^{\text{ext}}_{t+1} = R^{\text{ext}}$ for every t.

RETROAGENT (Figure 2) augments the training objective with an episode-level intrinsic reward R^{int} derived from hindsight self-reflection (Section 3.2). Like R^{ext}, this reward is assigned to every

action step, yielding $\mathcal{J}_{\text{RetroAgent}}(\theta) = \mathbb{E}_\theta \left[\sum_{t=0}^{T-1} \gamma^t \left(R^{\text{ext}} + R^{\text{int}}\right)\right]$, where $\mathbb{E}_\theta$ averages over task sampling $x \sim \mathcal{D}$, trajectory generation $\tau \sim \Pi_\theta(\cdot \mid x, \mathcal{M})$, and any stochasticity in self-reflection, conditional on the current memory buffer $\mathcal{M}$. Here, Π_θ denotes a mixture of trajectory distributions induced jointly by the environment and either the base policy, prompted with x, or the memory-augmented policy, prompted with $f_{\text{memory}}(x, \mathcal{M})$. The function f_{memory} augments x with a memory entry selected by our SIMUTIL-UCB retrieval strategy based on relevance and utility (Section 3.3). The buffer $\mathcal{M}$ grows as the agent accumulates experience.

Hindsight Self-Reflection. RETROAGENT learns from experience by reflecting on each completed episode. Given a trajectory τ, the reflection function produces $z = f_{\text{reflect}}(\tau) = (\phi_{(x,\tau)}, c, l)$,[2] where: (i) the *potential score* $\phi_{(x,\tau)} \in [0, 1]$ estimates the subtask completion rate and determines the intrinsic numerical reward R^{int} (Section 3.2); (ii) the *success prediction* $c \in \{\text{success}, \text{failure}\}$ assesses the episode's outcome; and (iii) the natural-language *retrospective lesson* l summarizes insights from the trajectory. The lesson is stored in a memory entry $m \in \mathcal{M}$ and later retrieved through $f_{\text{memory}}(x, \mathcal{M})$ to provide intrinsic language feedback (Section 3.3). To elicit high-quality feedback, we propose two variants.

In-Context Variant. We use *pairwise induction* through in-context learning (Wei et al., 2022), augmenting reflection with the current episode's external outcome indicator $I^{\text{ext}} \in \{\text{success}, \text{failure}\}$ and a reference trajectory τ_{ref} from an earlier training step with the opposite outcome: $z = f_{\text{reflect}}(\tau_{\text{ref}}, I^{\text{ext}}, \tau)$. Contrasting successful and failed trajectories is intended to reveal behavioral strengths and deficiencies, improving potential estimates and lessons (Lee et al., 2023). Here, c is an outcome assessment rather than an independent prediction, since I^{ext} is provided.

RL-Trained Variant. We jointly optimize decision-making and self-reflection. The reflection policy φ_θ generates $z = (\phi_{(x,\tau)}, c, l)$ conditioned on τ, without observing I^{ext}. We reward correct outcome predictions using $R^{\text{reflect}} := |R^{\text{ext}}_{\text{max}}| \, \mathbf{1}\{c = I^{\text{ext}}\}$, where $\mathbf{1}\{\cdot\}$ is the indicator function. The constant $|R^{\text{ext}}_{\text{max}}|$, the magnitude of the maximum terminal extrinsic reward, sets the reward scale.[3] This reward directly supervises c, not the potential score or lesson.

The composite objective is:

$$\mathcal{J}_{\text{RetroAgent}}(\theta) = \underbrace{\mathbb{E}_{\tau \sim \Pi_\theta(\cdot|x,\,\mathcal{M}) \times P}\left[\sum_{t=0}^{T-1} \gamma^t \left(R^{\text{ext}} + R^{\text{int}}\right)\right]}_{\text{Decision-Making}} + \underbrace{\lambda_{\text{reflect}} \cdot \mathbb{E}_{z \sim \varphi_\theta(\cdot|\tau)}\left[R^{\text{reflect}}\right]}_{\text{Self-Reflection}}, \quad (1)$$

where R^{ext}_t and R^{int}_t are the rewards at step t, and $\lambda_{\text{reflect}} \geq 0$ controls the reflection term's weight. Setting $\lambda_{\text{reflect}} = 0$ recovers Equation 3.1. Prompt templates and optimization details appear in Appendix Q and Section 3.4, respectively.

3.2 INTRINSIC NUMERICAL FEEDBACK FOR DIRECTED EXPLORATION

Sparse terminal rewards fail to distinguish uninformative failures from rollouts with meaningful partial progress. To guide exploration in vast action spaces, RETROAGENT converts the trajectory-level potential score $\phi_{(x,\tau)} \in [0, 1]$ into an intrinsic *capability-evolution* reward.

For each task x, we track a historical competence threshold $\Phi^{(k)}_x \in [0, 1]$ across iterations k, initialized to $\Phi^{(0)}_x = 0$. Given N rollouts at iteration k with binary success indicators $I^{\text{ext},(j)}_k \in \{0, 1\}$, we compute the empirical success rate $\bar{I}^{\text{ext}}_k(x) = \frac{1}{N}\sum_{j=1}^{N} I^{\text{ext},(j)}_k$ and update the baseline monotonically: $\Phi^{(k+1)}_x = \max\left\{\Phi^{(k)}_x, \bar{I}^{\text{ext}}_k(x)\right\}$. The intrinsic reward is then defined as the rectified gain over this baseline: $R^{\text{int}}_k(\tau) = \left[\phi_{(x,\tau),k} - \Phi^{(k)}_x\right]_+$, where $[a]_+ := \max\{a, 0\}$. While $\phi_{(x,\tau)}$ measures per-trajectory subtask completion and $\Phi^{(k)}_x$ reflects population-level success frequency, both are bounded in $[0, 1]$ and represent task competence (with 1.0 denoting full mastery). Subtracting

[2] For notational simplicity, we reuse τ to denote the interaction history of interleaved observations and actions.

[3] We leave alternative scaling strategies to future work.

the extrinsic success rate treats verified task performance as an adaptive capability hurdle: (i) On unsolved tasks ($\Phi_x \approx 0$), any intermediate progress ($\phi > 0$) earns intrinsic reward to kickstart exploration. (ii) As the agent masters a task ($\Phi_x \to 1$), the threshold rises, naturally suppressing intrinsic rewards for partial solutions the policy has already surpassed. Using the grounded success rate $\Phi_x^{(k)}$ rather than a running average of subjective potential scores prevents self-referential reward inflation. The total trajectory-level reward is $R_k^{\text{aug}}(\tau) = R^{\text{ext}}(\tau) + R_k^{\text{int}}(\tau)$.

3.3 Intrinsic Language Feedback via Reflection Memory

Numerical feedback identifies promising trajectories; language feedback provides reusable guidance on how to improve. RetroAgent connects retrospective reflection to future action through a retrieval-augmented memory. Its key design is SimUtil-UCB, which combines semantic relevance, outcome-based utility, and exploration of underused lessons.

Each entry is $m_i = (x_i, l_i, \tau_i, u_i, n_i, d_i)$, containing the source instruction, lesson, trajectory, estimated utility $u_i \in [0, 1]$, retrieval count $n_i \geq 0$, and source outcome $d_i \in \{\text{success}, \text{failure}\}$. We initialize $u_i = 0.5$ and $n_i = 0$, and cache instruction embeddings $\mathbf{v}_i = \mathcal{E}(x_i)$ using a frozen sentence encoder.

SimUtil-UCB retrieval. Given a task x, SimUtil-UCB ranks memories by semantic relevance, empirical utility, and an exploration bonus. Semantic relevance is measured by cosine similarity: $s_{\text{rel}}(x, x_i) = \frac{\mathcal{E}(x)^\top \mathbf{v}_i}{\|\mathcal{E}(x)\|_2 \|\mathbf{v}_i\|_2}$, with entries below 0.4 discarded. Each utility score is initialized to $u_i = 0.5$ and updated using an exponential moving average (EMA) (Klinker, 2011). Whenever a retrieved memory m_i is used in an episode with task success score $\hat{u}_t \in [0, 1]$, $u_i \leftarrow (1 - \beta_{\text{util}})u_i + \beta_{\text{util}}\hat{u}_t$, where $\beta_{\text{util}} \in (0, 1)$ controls the update rate. The EMA smooths episode-level variability while giving greater weight to recent outcomes: larger β_{util} enables faster adaptation, whereas smaller values retain more historical information. Utilities remain unchanged for memories not retrieved. To encourage exploration of less-used memories, we add a UCB bonus. Let n_i count retrievals of m_i and $N_{\mathcal{M}} = \sum_j n_j$. The utility-UCB value is $u_{\text{UCB}}^{(i)} = u_i + \kappa\sqrt{\frac{\ln \max(N_{\mathcal{M}}, 1)}{\max(n_i, 1)}}$, where the maxima handle zero counts and $\kappa = 1.0$ in our experiments. The final score is $S(m_i \mid x, \mathcal{M}) = \alpha s_{\text{rel}}(x, x_i) + (1 - \alpha)u_{\text{UCB}}^{(i)}$, where $\alpha \in [0, 1]$ balances relevance and utility-UCB. For the top-k eligible entries $\mathcal{K}$, we concatenate their lessons as $l_{\text{retrieved}} = \bigoplus_{i \in \mathcal{K}} l_i$ and construct $f_{\text{memory}}(x, \mathcal{M}) = x \oplus l_{\text{retrieved}}$. After retrieval, we increment n_i for each $i \in \mathcal{K}$.

3.4 Policy Optimization with Dual Intrinsic Feedback

RetroAgent supports multiple RL algorithms; here, we adapt GRPO (Shao et al., 2024b) to multi-turn optimization with dual intrinsic feedback. We describe trajectory generation, decision-making optimization, and optional reflection training below.

Memory-Augmented Trajectory Generation. For each task $x \sim \mathcal{D}$, we generate N trajectories under $\Pi_{\theta_{\text{old}}}(\cdot \mid x, \mathcal{M}) \times P$ (Equation 3.1): half from the base policy $\pi_{\theta_{\text{old}}}(\cdot \mid x) \times P$ and half from the memory-augmented policy $\pi_{\theta_{\text{old}}}(\cdot \mid f_{\text{memory}}(x, \mathcal{M})) \times P$. Each trajectory $\tau^{(i)} = (s_0^{(i)}, a_0^{(i)}, \ldots, s_{T_i-1}^{(i)}, a_{T_i-1}^{(i)})$ contains T_i state–action pairs. This split balances experience reuse with independent exploration.

Decision-Making Objective. Since both R^{ext} and R^{int} are uniform across time steps (Section 3.1), the discounted return reduces to a trajectory-level scalar $G^{(i)} = \sum_{t=0}^{T_i-1} \gamma^t \big(R^{\text{ext},(i)} + R^{\text{int},(i)}\big)$, and every step within a trajectory shares the same group-relative advantage: $\hat{A}^{(i)} = \frac{G^{(i)} - \text{mean}\big(\{G^{(1)}, \ldots, G^{(N)}\}\big)}{\text{std}\big(\{G^{(1)}, \ldots, G^{(N)}\}\big)}$. Defining the per-token importance ratio as $\rho_{t,j}^{(i)}(\theta) = \frac{\pi_\theta(a_{t,j}^{(i)} \mid s_t^{(i)}, a_{t,<j}^{(i)})}{\pi_{\theta_{\text{old}}}(a_{t,j}^{(i)} \mid s_t^{(i)}, a_{t,<j}^{(i)})}$, the decision-making objective is formulated as:

$$\mathcal{J}_{\text{Decision-Making}}(\theta) = \mathbb{E}_{x \sim \mathcal{D},\, \{\tau^{(i)}\} \sim \Pi_{\theta_{\text{old}}}(\cdot \mid x, \mathcal{M}) \times P} \left[\frac{1}{N} \sum_{i=1}^{N} \frac{1}{T_i} \sum_{t=0}^{T_i-1} \sum_{j=1}^{|a_t^{(i)}|} \left(\mathcal{L}_{t,j}^{\text{clip}}(\theta,\, \hat{A}^{(i)}) - \right. \right.$$

$\beta\, D_{\mathrm{KL}}\big[\pi_\theta(\cdot \mid s_t^{(i)}) \,\|\, \pi_{\text{ref}}(\cdot \mid s_t^{(i)})\big]\Big)\Big]$, where $|a_t^{(i)}|$ denotes the number of tokens in action $a_t^{(i)}$. Following Liu et al. (2025a), we remove the length normalization term $\frac{1}{|a_t^{(i)}|}$. The clipped surrogate function is defined as $\mathcal{L}_{t,j}^{\text{clip}}\Big(\theta,\, \hat{A}^{(i)}\Big) = \min\Big(\rho_{t,j}^{(i)}(\theta)\,\hat{A}^{(i)},\ \text{clip}\Big(\rho_{t,j}^{(i)}(\theta),\, 1-\epsilon_{\text{clip}},\, 1+\epsilon_{\text{clip}}\Big)\,\hat{A}^{(i)}\Big)$, where ϵ_{clip} bounds the policy update and β controls the KL divergence regularization toward the reference policy π_{ref}. For the *in-context* self-reflection variant, the total objective is simply $\mathcal{J}_{\text{RetroAgent}}(\theta) = \mathcal{J}_{\text{Decision-Making}}(\theta)$.

Self-Reflection Objective (for RL-Trained Variant). The *RL-trained* variant also optimizes the reflection policy φ_θ. For each trajectory $\tau^{(i)}$, it generates $z^{(i)} = (\phi_{(x,\tau)}^{(i)}, c^{(i)}, l^{(i)})$, with success prediction $c^{(i)}$ scored by $R^{\text{reflect},(i)}$. We use REINFORCE (Williams, 1992): $\mathcal{J}_{\text{Self-Reflection}}(\theta) = \mathbb{E}_{\{z^{(i)}\sim\varphi_{\theta_{\text{old}}}(\cdot|\tau^{(i)})\}}\left[\frac{1}{N}\sum_{i=1}^{N} R^{\text{reflect},(i)} \sum_{j=1}^{|z^{(i)}|} \log \varphi_\theta\big(z_j^{(i)} \mid \tau^{(i)}, z_{<j}^{(i)}\big)\right]$, where $|z^{(i)}|$ is the reflection's token count. Gradients do not propagate through trajectory generation. The joint objective is $\mathcal{J}_{\text{RetroAgent}}(\theta) = \mathcal{J}_{\text{Decision-Making}}(\theta) + \lambda_{\text{reflect}}\mathcal{J}_{\text{Self-Reflection}}(\theta)$. Algorithm 1 in Appendix A summarizes RETROAGENT; Appendix M provides theoretical justifications.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Environments. We evaluate RETROAGENT on four tasks: (*i*) ALFWorld (Shridhar et al., 2021), text-based household tasks evaluated in seen (in-distribution, ID) and unseen (OOD) rooms; (*ii*) WebShop (Yao et al., 2022a), web-based shopping guided by user specifications; (*iii*) Sokoban (Racanière et al., 2017), box-pushing puzzles requiring long-horizon planning; and (*iv*) MineSweeper (Li et al., 2024), mine identification from numerical clues. We report success rate on all tasks and task score on WebShop. Environment configurations are provided in Appendix N.

Compared Methods. We compare RETROAGENT with four baseline categories, averaging results over three independent runs: (*i*) *Prompting-based methods*: ReAct (Yao et al., 2022c) and Reflexion (Shinn et al., 2023), which adds in-context self-reflection for iterative refinement. (*ii*) *RL algorithms*: RLOO (Kool et al., 2019; Ahmadian et al., 2024), GRPO (Shao et al., 2024b), and GiGPO (Feng et al., 2025), which uses anchor-state grouping for fine-grained credit assignment. (*iii*) *Memory-augmented and RL-based frameworks*: MemRL (Zhang et al., 2026), which updates memory with a frozen policy; EvolveR (Wu et al., 2025), which incorporates raw trajectories into optimization; Mem0 (Chhikara et al., 2025)+GRPO and SimpleMem (Liu et al., 2026a)+GRPO, which integrate persistent memory into training; SkillRL (Xia et al., 2026), which combines supervised fine-tuning and RL with teacher-induced skills; and GRPO with EMPG (Wang et al., 2025b), which uses entropy-modulated policy gradients. (*iv*) *Meta-RL* (Beck et al., 2025): LAMER (Jiang et al., 2025), which uses multi-episode training to encourage exploration and adaptation.

Implementation Details. We use Qwen-2.5-7B-Instruct (Qwen et al., 2025) and Llama-3.1-8B-Instruct (Grattafiori et al., 2024), with GRPO as the default RL algorithm. Agents follow the ReAct (Yao et al., 2022c) format, retrieving at test time lessons distilled from training trajectories. Training infrastructure, prompts, hyperparameters, and configurations are detailed in Appendix N.

4.2 MAIN RESULTS

Table 1 summarizes the main results. Our key findings are summarized below:

Dual intrinsic feedback improves policy learning. RETROAGENT outperforms all baselines across four benchmarks, exceeding GRPO by +14.4, +12.0, +21.4, and +8.6 points on ALFWorld, WebShop, Sokoban, and MineSweeper, respectively. On WebShop, it also surpasses GiGPO and SKILLRL, supporting the value of hindsight-generated feedback beyond extrinsic rewards.

Numerical and language feedback are complementary. RETROAGENT outperforms both memory-based methods (MemRL, EvolveR, SimpleMem+GRPO, and SKILLRL) and uncertainty-driven

Table 1: Main results on four benchmarks (mean ± standard deviation over three independent runs). All improvements are significant at $p < 0.01$. * marks results from prior works. Training frameworks use GRPO unless noted. "Success" and "Score" denote success rate and task score; w/ Teacher Model marks methods requiring a teacher model for skill induction.

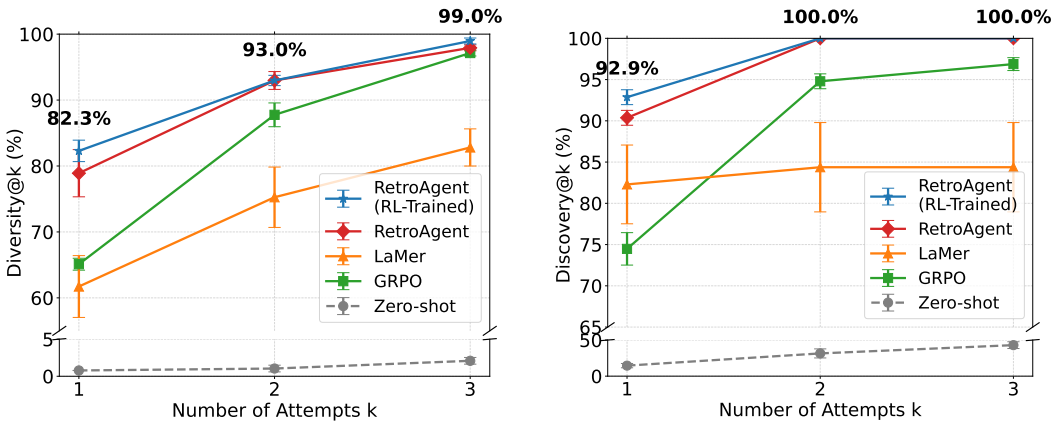
Method	ALFWorld	WebShop		Sokoban	MineSweeper
	Success (%)	Score (%)	Success (%)	Success (%)	Success (%)
Qwen-2.5-7B-Instruct (Zero-Shot)	$16.9_{\pm 1.8}$	$4.5_{\pm 1.8}$	$0.8_{\pm 0.0}$	$2.6_{\pm 0.5}$	$6.5_{\pm 1.6}$
Prompting-based Methods					
ReAct* (Yao et al., 2022c)	31.2	46.2	19.5	3.9	7.0
Reflexion* (Shinn et al., 2023)	42.7	58.1	28.8	4.3	7.4
Fine-tuning with RL					
RLOO* (Kool et al., 2019)	$75.5_{\pm 4.6}$	$80.3_{\pm 3.2}$	$65.7_{\pm 4.0}$	$9.9_{\pm 1.6}$	$32.8_{\pm 4.8}$
GRPO (Shao et al., 2024b)	$77.3_{\pm 4.3}$	$75.5_{\pm 3.6}$	$66.9_{\pm 1.2}$	$11.2_{\pm 2.5}$	$39.3_{\pm 2.7}$
GiGPO* (Feng et al., 2025)	$90.8_{\pm 1.3}$	$84.4_{\pm 2.9}$	$72.8_{\pm 3.2}$	$21.9_{\pm 2.8}$	$41.1_{\pm 1.2}$
Fine-tuning with RL-based Frameworks					
MemRL* (Zhang et al., 2026)	21.4	29.5	9.2	$4.2_{\pm 3.2}$	$7.0_{\pm 1.4}$
EvolveR* (Wu et al., 2025)	43.8	42.5	17.6	$6.0_{\pm 3.2}$	$11.7_{\pm 3.1}$
Mem0 (Chhikara et al., 2025)+GRPO*	54.7	58.1	37.5	–	–
SimpleMem (Liu et al., 2026a)+GRPO*	62.5	67.8	46.9	–	–
SKILLRL* (Xia et al., 2026) w/ Teacher Model	89.9	85.2	72.7	–	–
GRPO w/ EMPG* (Wang et al., 2025b)	78.5	81.0	69.3	$12.8_{\pm 2.3}$	$40.1_{\pm 3.6}$
Fine-tuning with Meta-RL Frameworks					
LAMER (Jiang et al., 2025)	$82.3_{\pm 3.6}$	–	$61.7_{\pm 4.7}$	$14.3_{\pm 1.2}$	$33.3_{\pm 1.8}$
RL Training with Extrinsic and Dual Intrinsic Feedback					
RETROAGENT (In-Context Reflection)	$91.7_{\pm 1.2}$	$87.6_{\pm 2.1}$	$78.9_{\pm 3.6}$	$32.6_{\pm 4.6}$	$47.9_{\pm 2.0}$
RETROAGENT (RL-Trained Reflection)	$\mathbf{95.6}_{\pm \mathbf{2.3}}$	$\mathbf{88.9}_{\pm \mathbf{1.3}}$	$\mathbf{82.3}_{\pm \mathbf{1.6}}$	$\mathbf{38.3}_{\pm \mathbf{3.4}}$	$\mathbf{48.2}_{\pm \mathbf{2.0}}$

GRPO w/ EMPG. These comparisons suggest complementary benefits from progress-aware exploration and language-based experience reuse.

Distilled lessons outperform raw trajectory reuse. On WebShop, RETROAGENT achieves 78.9–82.3% success, compared with EvolveR's 17.6%. This suggests that concise, transferable lessons may support future decisions better than raw trajectories containing task-specific or irrelevant details.

Training self-reflection yields further gains. Jointly optimizing the reflection policy raises success rates to 95.6%, 82.3%, and 38.3% on ALFWorld, WebShop, and Sokoban, respectively, suggesting that learned reflection improves feedback for policy optimization.

4.3 TEST-TIME ADAPTATION AND GENERALIZATION



(a) Test-time adaptation on WebShop (ID).

(b) Test-time adaptation on ALFWorld (OOD).

Figure 3: Test-time adaptation in an ID setting on WebShop and an OOD setting on ALFWorld.

Table 2: Impact of discounted returns and intrinsic reward shaping on GRPO. Capability-evolution rewards denote the intrinsic numerical feedback described in Section 3.2.

Method	Discounted Returns	Reward Type	WebShop	
			Task Score (%)	Success Rate (%)
GRPO (Baseline)	–	Extrinsic	$75.5_{\pm 3.6}$	$66.9_{\pm 1.2}$
GRPO	✓	Extrinsic	$84.2_{\pm 0.2}$	$74.7_{\pm 2.7}$
+ Progress-Guided Rewards	✓	Extrinsic	$84.2_{\pm 1.7}$	$75.0_{\pm 3.1}$
+ Capability-Evolution Rewards	✓	Extrinsic & Intrinsic	$\mathbf{88.2}_{\pm 2.1}$	$\mathbf{79.7}_{\pm 3.1}$

Test-Time Adaptation. Following Jiang et al. (2025), we evaluate Discovery@k (Hübotter et al., 2026), the probability of success within k attempts: Discovery@$k := P(\exists i \leq k : r(y_i \mid x) = 1)$.

RETROAGENT enables rapid test-time adaptation. From one to three attempts, discovery rises from 82.3% to 99.0% on in-distribution WebShop and from 92.9% to 100.0% on out-of-distribution ALFWorld (Figure 3). RETROAGENT consistently outperforms LAMER, with a widening OOD margin as k increases, suggesting more effective use of repeated attempts.

RETROAGENT internalizes intrinsic feedback during training. Removing memory retrieval only slightly reduces Discovery@1 and Discovery@2 (*e.g.,* 78.9% → 76.8% for in-context self-reflection), leaving Discovery@3 unchanged (Table 4, Appendix B). This suggests that training internalizes benefits of intrinsic feedback, reducing reliance on inference-time memory access. Appendix B reports generalization to more challenging tasks.

4.4 EFFECT OF INTRINSIC NUMERICAL FEEDBACK

We defer the analysis of in-context self-reflection to Appendix C due to space constraints. We evaluate discounted returns and intrinsic reward shaping on GRPO (Table 2; Figure 7a, Appendix D). Our *progress-guided* baseline replaces $\phi_{(x,\tau)}$ in Section 3 with binary success I^{Ext}, retaining the same rectified-gain formulation. Our contribution is not rewarding partial progress, as in dense rewards and progress-based shaping, but estimating cross-attempt progress through retrospective self-reflection without predefined subtask verifiers or dense annotations. This uses the agent's reflector as a reward estimator, rather than a separately trained reward model. Unlike standard potential-based shaping, our rectified cross-attempt gain carries no claimed policy-invariance guarantee.

Intrinsic numerical feedback improves policy optimization. On WebShop, discounted returns for trajectory-level advantages improve GRPO by +8.7 percentage points in task score and +7.8 in success rate. Adding capability-evolution rewards raises these metrics to 88.2% and 79.7%, respectively, with consistent gains from step 25 onward. Their advantage over the binary-outcome baseline supports the value of self-assessed progress within this formulation, not superiority over reward-shaping methods generally.

4.5 EFFECT OF INTRINSIC LANGUAGE FEEDBACK

We now isolate the role of retrieval by comparing SIMUTIL-UCB with similarity-only and similarity–utility retrieval. All variants use half-group memory augmentation (Table 7, Appendix D).

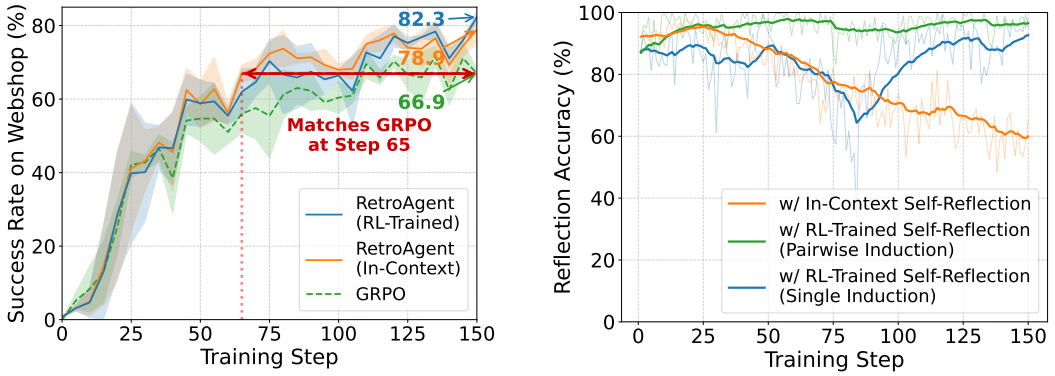
Balancing relevance, utility, and exploration is critical. Although discounted returns improve GRPO, adding similarity-only or similarity–utility retrieval reduces performance. In contrast, SIMUTIL-UCB improves performance to 86.4% task score and 78.6% success rate. Similarity-only retrieval benefits GRPO without discounted returns (Table 6, Appendix C), suggesting that the effectiveness of memory retrieval depends on return estimation. Access-frequency analysis further shows that SIMUTIL-UCB distributes retrievals more evenly than similarity-only retrieval. Together, these results support combining relevance and utility with exploration to limit reliance on a narrow subset of memories. Appendix D reports validation dynamics and retrieval statistics.

4.6 EFFECT OF COMBINING DUAL INTRINSIC FEEDBACK

Table 3 evaluates dual feedback, and Figure 4 compares in-context and RL-trained reflection.

Table 3: Individual and combined effects of intrinsic numerical and language feedback under different self-reflection mechanisms on WebShop. Rows above the dashed line ablate each feedback type in isolation; rows below combine both (Dual).

Method	Intrinsic Feedback	Self-Reflection Mechanism	WebShop	
			Task Score (%)	Success Rate (%)
GRPO (Baseline)	–	–	$75.5_{\pm 3.6}$	$66.9_{\pm 1.2}$
+ Capability-Evolution Rewards	Numerical	Pairwise Induction	$88.2_{\pm 2.1}$	$79.7_{\pm 3.1}$
+ SimUtil-UCB Memory Retrieval	Language	Pairwise Induction	$86.4_{\pm 1.8}$	$78.6_{\pm 1.6}$
RetroAgent (In-Context)	Dual	Pairwise Induction	$87.6_{\pm 2.1}$	$78.9_{\pm 3.6}$
RetroAgent (RL-Trained)	Dual	Pairwise Induction	$87.0_{\pm 1.4}$	$77.1_{\pm 1.0}$
RetroAgent (RL-Trained)	Dual	Single Induction	$\mathbf{88.9}_{\pm \mathbf{1.3}}$	$\mathbf{82.3}_{\pm \mathbf{1.6}}$



(a) Valid-set performance over the course of training.

(b) Reflection accuracy smoothed with EMA.

Figure 4: In-context *vs.* RL-trained self-reflection mechanisms in RetroAgent on WebShop.

Jointly trained dual feedback improves performance. With RL-trained reflection, RetroAgent improves success rate by approximately 3 percentage points over either capability-evolution rewards or SimUtil-UCB retrieval alone. In-context dual feedback slightly underperforms numerical rewards alone, suggesting possible interference without joint optimization.

RL training stabilizes reflection accuracy. In-context accuracy declines as the policy improves despite access to extrinsic success signals, whereas RL-trained accuracy recovers from an early dip before step 75 and remains more stable (Figure 4b). The dip may reflect reflection adapting more slowly than the policy; the initial accuracy gap is consistent with less informative single versus pairwise induction (Appendix C). Conditioning the trained reflector on a reference trajectory, $z = f_{\text{reflect}}(\tau_{\text{ref}}, \tau)$, achieves the highest reflection accuracy without improving task performance (Table 3). This supports single induction for our setting and suggests that pairwise comparison may favor relative outcome inference over standalone evaluation. Further analyses cover exploration (Appendix E), training efficiency (Appendix H), coefficient sensitivity (Appendices G), cross-architecture generalization (Appendix I), model-size scaling (Appendix J), and qualitative examples (Appendix K).

5 Conclusion

We present RetroAgent, an online RL framework for cross-episode policy improvement through retrospective self-reflection. It combines extrinsic task-success rewards with dual intrinsic feedback: numerical rewards that use self-assessed progress relative to prior attempts as a proxy for capability improvement, and language feedback that distills reusable lessons into memory. SimUtil-UCB retrieves lessons by balancing semantic relevance, empirical utility, and exploration of under-tested memories. Together, these mechanisms support productive exploration and experience reuse without predefined subtask verifiers or dense reward annotations. Across four agentic benchmarks, RetroAgent achieves SOTA results in the evaluated settings, with strong test-time adaptation and OOD generalization. Future work will improve joint optimization of reflection and decision-making and extend the framework to multi-agent and open-ended settings.

AI USE STATEMENT

We used AI tools to improve the clarity and wording of the manuscript and to assist in evaluating the quality of the trained agent's responses. Details of the AI-assisted evaluation are provided in Appendix C.

ETHICS STATEMENT

This research adheres to the ICLR Code of Ethics. All experiments are conducted on publicly available benchmarks—ALFWorld, WebShop, Sokoban, and Minesweeper—in simulated, sandboxed environments, without human subjects, personally identifiable information, or private data. The proposed intrinsic feedback mechanisms—capability-evolution rewards and memory-based language feedback—operate solely within these environments and do not interact with real users or real-world systems. We use open-source base models to support transparency and reproducibility. Although this experimental setting limits direct risks, it does not establish the safety or fairness of real-world deployment, which would require further evaluation.

REPRODUCIBILITY STATEMENT

We describe the proposed RETROAGENT framework in Section 3, provide detailed implementation information in Appendix N, and document the experimental prompts in Appendix Q.

REFERENCES

David Abel, André Barreto, Benjamin Van Roy, Doina Precup, Hado P van Hasselt, and Satinder Singh. A definition of continual reinforcement learning. *Advances in Neural Information Processing Systems*, 36:50377–50407, 2023.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Anthropic. Introducing agent skills. *Claude Blog*, 2025.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A tutorial on meta-reinforcement learning. *Foundations and Trends in Machine Learning*, 18(2-3):224–384, 2025.

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. Reinforcement learning for long-horizon interactive llm agents, 2025. URL https://arxiv.org/abs/2502.01600.

Yen-Chi Chen. A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology*, 1(1):161–187, 2017.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory, 2025. URL https://arxiv.org/abs/2504.19413.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory, 2026. URL https://arxiv.org/abs/2508.06433.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=QXEhBMNrCW.

Dan Friedman and Adji Bousso Dieng. The vendi score: A diversity evaluation metric for machine learning. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Jingtong Gao, Ling Pan, Yejing Wang, Rui Zhong, Chi Lu, Qingpeng Cai, Peng Jiang, and Xiangyu Zhao. Navigate the unknown: Enhancing llm reasoning with intrinsic motivation guided exploration, 2025. URL https://arxiv.org/abs/2505.17621.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International conference on machine learning*, pp. 10835–10866. PMLR, 2023.

Anirudh Goyal, Abram Friesen, Andrea Banino, Theophane Weber, Nan Rosemary Ke, Adria Puigdomenech Badia, Arthur Guez, Mehdi Mirza, Peter C Humphreys, Ksenia Konyushova, et al. Retrieval-augmented reinforcement learning. In *International Conference on Machine Learning*, pp. 7740–7765. PMLR, 2022.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas

Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Geoffrey E Hinton, Steven J Nowlan, et al. How learning can guide evolution. *Complex systems*, 1 (3):495–502, 1987.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation, 2026. URL https://arxiv.org/abs/2601.20802.

Yulun Jiang, Liangze Jiang, Damien Teney, Michael Moor, and Maria Brbic. Meta-rl induces exploration in language agents, 2025. URL https://arxiv.org/abs/2512.16848.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. Understanding the effects of RLHF on LLM generalisation and diversity. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=PXD3FAVHJT.

Frank Klinker. Exponential moving average versus moving exponential average. *Mathematische Semesterberichte*, 58(1):97–107, 2011.

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 REINFORCE samples, get a baseline for free!, 2019. URL https://openreview.net/forum?id=r1lgTGL5DE.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Ren Lu, Thomas Mesnard, Johan Ferret, Colton Bishop, Ethan Hall, Victor Carbune, and Abhinav Rastogi. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. 2023.

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: Human Language Technologies (Volume 1: Long Papers)*, pp. 59–81, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3):293–321, 1992.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for llm agents, 2026a. URL https://arxiv.org/abs/2601.02553.

Tennison Liu and Mihaela van der Schaar. Position: Truly self-improving agents require intrinsic metacognitive learning. In *Forty-second International Conference on Machine Learning Position Paper Track*, 2025. URL https://openreview.net/forum?id=4KhDd0Ozqe.

Zeyuan Liu, Jeonghye Kim, Xufang Luo, Dongsheng Li, and Yuqing Yang. Exploratory memory-augmented LLM agent via hybrid on- and off-policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=UOzxviKVFO.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. In *Second Conference on Language Modeling*, 2025a. URL https://openreview.net/forum?id=5PAF7PAY2Y.

Zichen Liu, Anya Sims, Keyu Duan, Changyu Chen, Simon Yu, Xiangxin Zhou, Haotian Xu, Shaopan Xiong, Bo Liu, Chenmien Tan, et al. Gem: A gym for agentic llms. *arXiv preprint arXiv:2510.01051*, 2025b.

Kristen E Lyons and Philip David Zelazo. Monitoring, metacognition, and executive function: Elucidating the role of self-reflection in the development of self-regulation. *Advances in child development and behavior*, 40:379–412, 2011.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko,

Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. Gpt-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Ramesh Motwani, Elan Sopher Markowitz, Julia Kiseleva, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent q: Advanced reasoning and learning for autonomous AI agents, 2025. URL https://openreview.net/forum?id=LuytzzohTa.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Banafsheh Rafiee and Richard Sutton. Toward enactive artificial intelligence. *arXiv preprint arXiv:2605.24238*, 2026.

Jette Randløv and Preben Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *ICML*, volume 98, pp. 463–471, 1998.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

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

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

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. *Advances in neural information processing systems*, 35:9460–9471, 2022.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7584–7600, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.409. URL https://aclanthology.org/2024.acl-long.409/.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Fahim Tajwar, Yiding Jiang, Abitha Thankaraj, Sumaita Sadia Rahman, J Zico Kolter, Jeff Schneider, and Russ Salakhutdinov. Training a generally curious agent. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=UeB3Hdrhda.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. *arXiv preprint arXiv:2407.18901*, 2024.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024a. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

Hanlin Wang, Chak Tou Leong, Jiashuo Wang, Jian Wang, and Wenjie Li. Spa-rl: Reinforcing llm agents via stepwise progress attribution, 2025a. URL https://arxiv.org/abs/2505.20732.

Jiawei Wang, Jiacai Liu, Yuqian Fu, Yingru Li, Xintao Wang, Yuan Lin, Yu Yue, Lin Zhang, Yang Wang, and Ke Wang. Harnessing uncertainty: Entropy-modulated policy gradients for long-horizon llm agents, 2025b. URL https://arxiv.org/abs/2509.09265.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=O0nBMRlkc8.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024c.

Sai Wang, Yu Wu, and Zhongwen Xu. Cogito, ergo ludo: An agent that learns to play by reasoning and planning, 2025c. URL https://arxiv.org/abs/2509.25052.

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

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*, 2022, 2022. URL https://openreview.net/forum?id=yzkSU5zdwD.

Quan Wei, Siliang Zeng, Chenliang Li, William Brown, Oana Frunza, Wei Deng, Anderson Schneider, Yuriy Nevmyvaka, Yang Katie Zhao, Alfredo Garcia, and Mingyi Hong. Reinforcing multi-turn reasoning in llm agents via turn-level reward design, 2025. URL https://arxiv.org/abs/2505.11821.

Jiayi Weng. Learning beyond gradients. https://trinkle23897.github.io/learning-beyond-gradients/, May 2026. Blog post.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, and Botian Shi. Evolver: Self-evolving llm agents through an experience-driven lifecycle, 2025. URL https://arxiv.org/abs/2510.16079.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Xin Guo, Dingwen Yang, Chenyang Liao, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. AgentGym: Evaluating and training large language model-based agents across diverse environments. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27914–27961, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1355. URL https://aclanthology.org/2025.acl-long.1355/.

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, 2026. URL https://arxiv.org/abs/2602.08234.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022a. URL http://papers.nips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Shunyu Yao, Howard Chen, John Yang, and Karthik R Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022b. URL https://openreview.net/forum?id=R9KnuFlvnU.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022c.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh R N, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, Ran Xu, Phil L Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. Retroformer: Retrospective large language agents with policy gradient optimization. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=KOZu91CzbK.

Chaoyun Zhang, Liqun Li, Shilin He, Xu Zhang, Bo Qiao, Si Qin, Minghua Ma, Yu Kang, Qingwei Lin, Saravan Rajmohan, et al. Ufo: A ui-focused agent for windows os interaction. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 597–622, 2025a.

Hanchen Zhang, Xiao Liu, Bowen Lv, Xueqiao Sun, Bohao Jing, Iat Long Iong, Zhenyu Hou, Zehan Qi, Hanyu Lai, Yifan Xu, Rui Lu, Hongning Wang, Jie Tang, and Yuxiao Dong. Agentrl: Scaling agentic reinforcement learning with a multi-turn, multi-task framework, 2025b. URL https://arxiv.org/abs/2510.04206.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory, 2026. URL https://arxiv.org/abs/2601.03192.

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*, 2025c.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and Jun Wang. Memento: Fine-tuning llm agents without fine-tuning llms, 2025. URL https://arxiv.org/abs/2508.16153.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=b6rA0kAHT1.

CONTENTS

1 Introduction **1**

2 Related Work **2**

3 RETROAGENT **3**

3.1 General Overview 3

3.2 Intrinsic Numerical Feedback for Directed Exploration 4

3.3 Intrinsic Language Feedback via Reflection Memory 5

3.4 Policy Optimization with Dual Intrinsic Feedback 5

4 Experiments **6**

4.1 Experimental Setup 6

4.2 Main Results 6

4.3 Test-Time Adaptation and Generalization 7

4.4 Effect of Intrinsic Numerical Feedback 8

4.5 Effect of Intrinsic Language Feedback 8

4.6 Effect of Combining Dual Intrinsic Feedback 8

5 Conclusion **9**

A The RETROAGENT Algorithm **21**

B Test-Time Generalization to Challenging Tasks **22**

C Analysis of In-Context Self-Reflection **22**

D Additional Analysis of Intrinsic Numerical and Language Feedback **23**

E Effect of Intrinsic Feedback on Exploration **24**

F Sensitivity to the Relevance–Utility Tradeoff Coefficient **26**

G Sensitivity to the Self-Reflection Objective Weight **26**

H Training Efficiency **26**

I Generalization Across Model Architectures **27**

J Scaling Across Model Sizes **27**

K Qualitative Analysis **27**

L Extended Related Work **29**

M Theoretical Analysis: Synergistic Benefit of Dual Feedback **30**

N Implementation Details **33**

O Superiority of Pairwise Induction over Single Induction **35**

P Generated Trajectories by RETROAGENT **35**

Q Task Prompts **38**

Q.1 Prompt Templates for In-Context Self-Reflection 38

Q.2 Prompt Templates for RL-Trained Self-Reflection 44

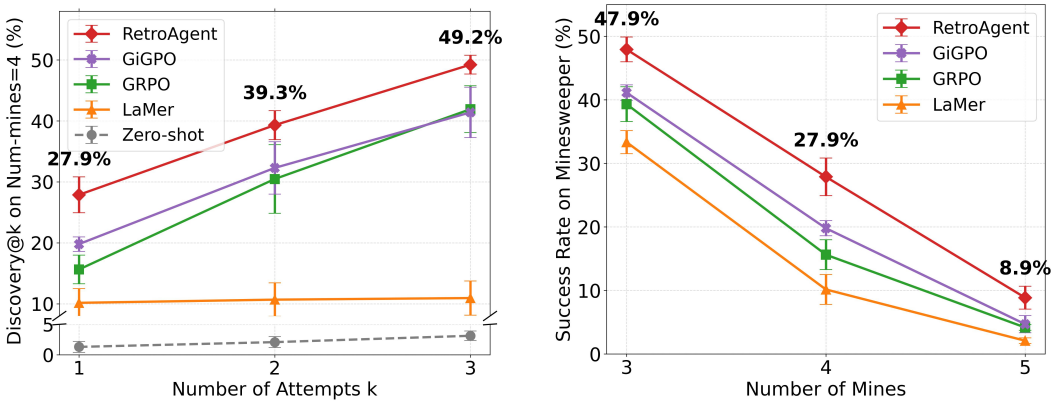
Q.3 Prompts for Analyzing the Quality of Intrinsic Feedback 50

A THE RETROAGENT ALGORITHM

Algorithm 1 RETROAGENT Training Framework

Require: Training dataset $\mathcal{D}$, Sentence encoder $\mathcal{E}$, hyperparameters $\alpha, \beta_{\text{util}}, \kappa, \lambda_{\text{reflect}}$.

Initialize memory buffer $\mathcal{M} \leftarrow \emptyset$, policy parameters θ, historical baselines $\Phi_x \leftarrow 0$ for all $x \in \mathcal{D}$.
for each training iteration **do**
  Sample task instruction $x \sim \mathcal{D}$.
  % 1. Memory Retrieval (SIMUTIL-UCB)
  if $\mathcal{M} \neq \emptyset$ **then**
    Compute semantic relevance s_{rel} via cosine similarity using $\mathcal{E}(x)$.
    Retrieve top-k entries $\mathcal{K}$ maximizing $S(m_i \mid x, \mathcal{M}) = \alpha s_{\text{rel}} + (1-\alpha) u^{(i)}_{\text{util-UCB}}$.
    Form augmented input $f_{\text{memory}}(x, \mathcal{M}) = x \oplus l_{\text{retrieved}}$.
    Increment access counts: $n_i \leftarrow n_i + 1$ for all $i \in \mathcal{K}$.
  end if
  % 2. Trajectory Generation
  Generate $N/2$ trajectories τ via base policy $\pi_{\theta_{\text{old}}}(\cdot \mid x)$.
  Generate $N/2$ trajectories τ via memory-augmented policy $\pi_{\theta_{\text{old}}}\big(\cdot \mid f_{\text{memory}}(x, \mathcal{M})\big)$.
  % 3. Self-Reflection & Intrinsic Feedback
  for each trajectory $\tau^{(i)}$ in the N rollouts **do**
    Observe extrinsic reward $R^{\text{ext},(i)}$ and outcome $I^{\text{ext},(i)}$.
    Generate reflection tuple $z^{(i)} = (\phi^{(i)}_{(x,\tau)}, c^{(i)}, l^{(i)})$ via reflection function f_{reflect} or policy φ_θ.
    Compute capability-evolution intrinsic reward: $R^{\text{int},(i)} \leftarrow \max\big(0,\ \phi^{(i)}_{(x,\tau)} - \Phi_x\big)$.
    Store new memory entry $m^{(i)}$ into buffer $\mathcal{M}$.
  end for
  Update task baseline: $\Phi_x \leftarrow \max\big(\Phi_x,\ \frac{1}{N}\sum_{j=1}^{N} I^{\text{ext},(j)}\big)$.
  Update utilities u_i for retrieved entries $i \in \mathcal{K}$ via EMA: $u_i \leftarrow (1-\beta_{\text{util}})u_i + \beta_{\text{util}}\hat{u}_t$.
  % 4. Policy Optimization (Dual Feedback)
  Compute advantages $\hat{A}^{(i)}$ using composite returns $G^{(i)} = \sum \gamma^t \big(R^{\text{ext},(i)} + R^{\text{int},(i)}\big)$.
  Update decision-making policy θ by maximizing $\mathcal{J}_{\text{Decision-Making}}(\theta)$ via GRPO.
  if using RL-Trained Reflection Variant **then**
    Compute reflection reward: $R^{\text{reflect},(i)} \leftarrow R^{\text{ext},(i)} \cdot \mathbf{1}\{c = I^{\text{ext}}\}$.
    Update reflection policy φ_θ by maximizing $\mathcal{J}_{\text{Self-Reflection}}(\theta)$ via REINFORCE.
  end if
end for



(a) Test-time adaptation using Discovery@k on harder instances (trained with 3 mines, evaluated with 4 mines).

(b) Generalization across increasing difficulty levels (evaluated with the number of mines ranging from 3 to 5).

Figure 5: Robustness to challenging tasks on MineSweeper.

B Test-Time Generalization to Challenging Tasks

Table 4: Impact of memory retrieval on test-time adaptation.

Method	Memory Retrieval	WebShop		
		Discovery@1 (%)	Discovery@2 (%)	Discovery@3 (%)
GRPO (Baseline)	–	$66.9_{\pm 1.2}$	$87.8_{\pm 1.8}$	$97.1_{\pm 0.5}$
RetroAgent (In-Context)	×	$76.8_{\pm 1.6}$	$91.9_{\pm 1.2}$	$98.4_{\pm 0.0}$
RetroAgent (RL-Trained)	×	$77.1_{\pm 1.6}$	$91.7_{\pm 1.2}$	$99.0_{\pm 0.5}$
RetroAgent (In-Context)	✓	$78.9_{\pm 3.6}$	$93.0_{\pm 1.4}$	$97.9_{\pm 0.5}$
RetroAgent (RL-Trained)	✓	$\mathbf{82.3}_{\pm \mathbf{1.6}}$	$\mathbf{93.0}_{\pm \mathbf{0.8}}$	$\mathbf{99.0}_{\pm \mathbf{0.5}}$

Robustness to Challenging Tasks. Following Jiang et al. (2025), we assess robustness on MineSweeper under two harder-than-training settings (Figure 5): (i) increasing the mine count from 3 to 4, and (ii) varying the mine count from 3 to 5.

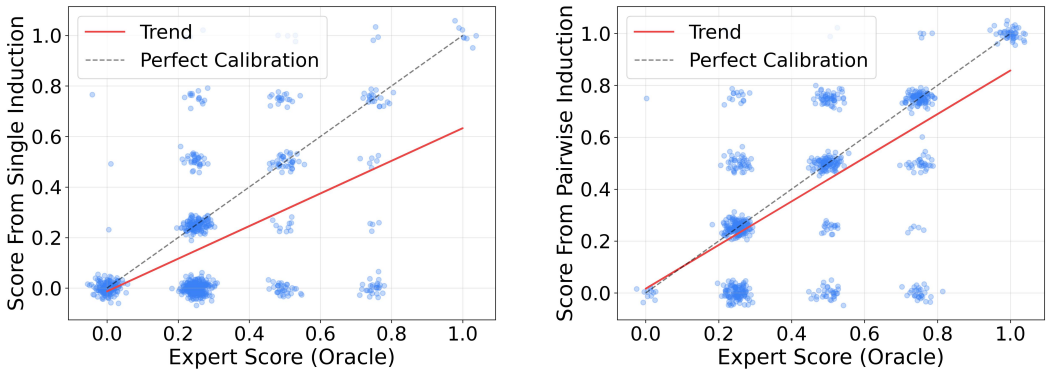
RetroAgent remains robust under increased difficulty. RetroAgent consistently outperforms all baselines in both settings, showing rapid adaptation to harder instances (Figure 5a) and graceful degradation as task difficulty increases (Figure 5b).

C Analysis of In-Context Self-Reflection

Table 5: Quality of lessons (*i.e.,* memories) generated via single-trajectory *vs.* pairwise-trajectory induction, as assessed by GPT-4o.

Method	Hallucination Rate (%)		Estimated Utility Score (%)					
			Failure			Success		
	Failure (↓)	Success (↓)	Low (↓)	Med (−)	High (↑)	Low (↓)	Med (−)	High (↑)
Single Induction	8.8	15.1	8.8	78.2	12.9	12.2	75.6	12.2
Pairwise Induction	**3.8**	**11.9**	**3.1**	76.7	**20.1**	**6.2**	76.2	**17.6**

The effectiveness of RetroAgent depends on the quality of self-reflection, which determines both the accuracy of intrinsic numerical feedback and the utility of intrinsic language feedback.



(a) Completion scores via single induction.

(b) Completion scores via pairwise induction.

Figure 6: Accuracy of subtask completion scores generated via single-trajectory (single) *vs.* pairwise-trajectory (pairwise) induction for Qwen-2.5-7B-Instruct on WebShop.

Table 6: Effect of induction method and augmentation ratio on GRPO performance. Augmentation Ratio denotes the fraction of sampled trajectories per prompt that receive memory-augmented generation; the remaining trajectories are sampled without augmentation.

Method	**Augmentation Ratio**	**WebShop**	
		Task Score (%)	Success Rate (%)
GRPO	—	$75.5_{\pm 3.6}$	$66.9_{\pm 1.2}$
+ Single Induction	100% (Full Group)	$81.3_{\pm 2.6}$	$70.3_{\pm 2.1}$
+ Pairwise Induction	100% (Full Group)	$82.3_{\pm 1.3}$	$72.9_{\pm 1.6}$
+ Pairwise Induction	50% (Half Group)	$\mathbf{82.4}_{\pm 2.9}$	$\mathbf{75.3}_{\pm 4.3}$

We compare single-trajectory and pairwise-trajectory induction under the in-context self-reflection mechanism.

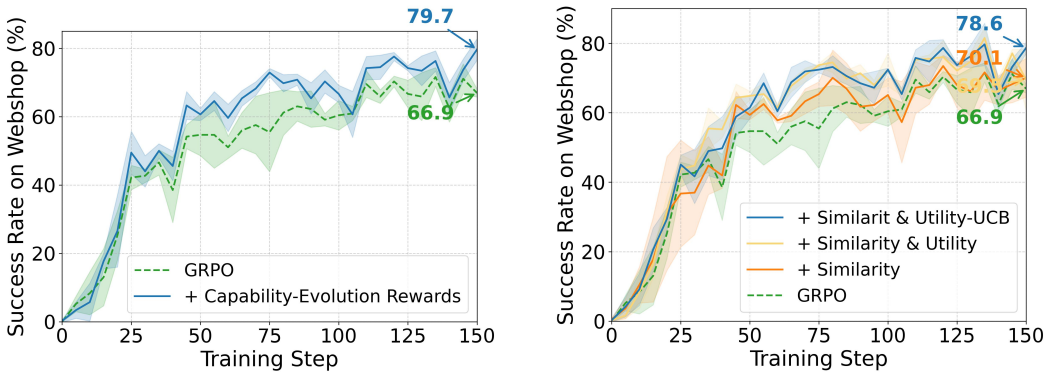
For numerical feedback, we use GPT-4o subtask completion scores (OpenAI et al., 2024) as oracle estimates and measure correlation with each induction method. For language feedback, we use GPT-4o to evaluate lesson quality. We further assess downstream impact by augmenting GRPO with lessons from each method, retrieved by semantic relevance to the task prompt (Table 6). Details are provided in Appendix O.

Pairwise induction yields more accurate and useful self-reflection. Pairwise-trajectory induction produces intrinsic numerical feedback that better matches oracle subtask completion scores, as shown by its closer alignment with the dashed oracle curve in Figure 6. It also generates higher-quality language feedback, with lower hallucination rates and higher estimated utility (Table 5). These improvements translate into better downstream optimization: GRPO augmented with pairwise-induction lessons outperforms the single-induction variant (72.9% vs. 70.3%; Table 6).

Preserving unguided exploration prevents premature convergence. Table 6 shows that half-group memory augmentation outperforms full-group augmentation (75.3% vs. 72.9% success). This indicates that applying memory-guided generation to all sampled trajectories reduces diversity and can promote premature convergence to suboptimal strategies.

D Additional Analysis of Intrinsic Numerical and Language Feedback

Having shown in Appendix C that intrinsic language feedback improves policy optimization, we evaluate SimUtil-UCB against two ablations: similarity-based retrieval (Criterion 1 only) and similarity–utility retrieval (Criteria 1–2, without the exploration bonus). Table 7 reports final per-



(a) Impact of capability-evolution rewards.

(b) Impact of memory-retrieval strategies.

Figure 7: Valid-set performance dynamics on WebShop when augmenting GRPO with intrinsic numerical feedback (a) or intrinsic language feedback (b).

Table 7: Impact of intrinsic language feedback on GRPO using different memory-retrieval strategies. SimUtil-UCB denotes the our proposed memory retrieval strategy (Section 3.3).

Method	Discounted Returns	Retrieval Strategy	WebShop	
			Task Score (%)	Success Rate (%)
GRPO (Baseline)	–	–	$75.5_{\pm 3.6}$	$66.9_{\pm 1.2}$
GRPO	✓	–	$84.2_{\pm 0.2}$	$74.7_{\pm 2.7}$
+ Memory Retrieval	✓	Similarity	$79.1_{\pm 7.1}$	$70.1_{\pm 5.5}$
+ Memory Retrieval	✓	Similarity & Utility	$78.4_{\pm 11.4}$	$69.5_{\pm 8.7}$
+ Memory Retrieval	✓	SimUtil-UCB	$\mathbf{86.4 \pm 1.8}$	$\mathbf{78.6 \pm 1.6}$

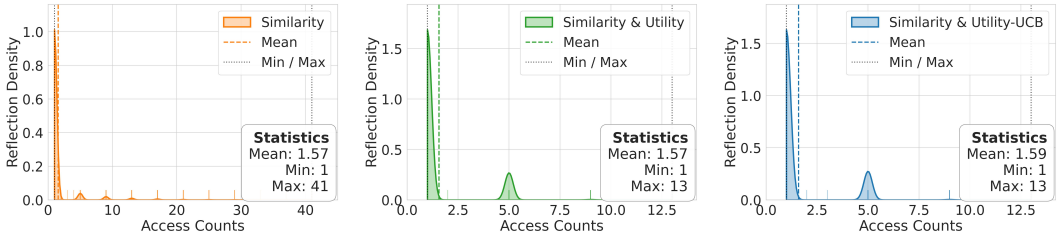
formance, and Figure 7b shows validation dynamics. All experiments use half-group memory augmentation.

Balancing relevance, utility, and exploration is critical. As shown in Table 7, discounted returns improve GRPO, but adding memories retrieved by similarity-only or similarity–utility criteria degrades performance. This contrasts with Table 6, where similarity-based retrieval improves standard GRPO without discounted returns, suggesting that discounted returns may amplify low-quality memory-guided behaviors. In contrast, SIMUTIL-UCB consistently improves performance, reaching an 86.4% task score and a 78.6% success rate (Table 7; Figure 7b). By adding the UCB exploration bonus, SIMUTIL-UCB avoids over-reliance on semantically similar or historically high-utility memories and promotes broader use of under-explored lessons.

Figure 8 further analyzes retrieval frequency. Each memory instance is initialized with count 1, which increments after retrieval. SIMUTIL-UCB distributes access more evenly, with most instances retrieved about 5 times (Figure 8c), whereas similarity-based retrieval concentrates on a small subset, with many instances exceeding 15 retrievals (Figure 8a). This confirms that the UCB bonus diversifies memory usage and contributes to stronger final performance.

E Effect of Intrinsic Feedback on Exploration

Both intrinsic numerical and language feedback are designed to improve RL by guiding exploration: capability-evolution rewards encourage promising action sequences, while retrieved lessons discourage repeated failures and reinforce effective strategies. We test this hypothesis on the WebShop test set under three settings: (i) GRPO with capability-evolution rewards only, (ii) GRPO with SimUtil-UCB memory retrieval only, and (iii) RETROAGENT with in-context or RL-trained self-reflection. We quantify trajectory diversity using the Vendi Score (Friedman & Dieng, 2023) over both successful and failed trajectories.



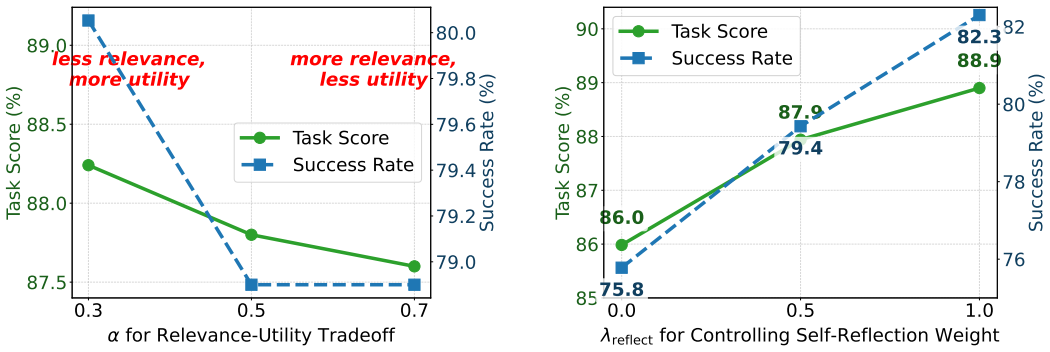
(a) Similarity-based retrieval. (b) Similarity & utility. (c) SimUtil-UCB.

Figure 8: Distribution of accumulated memory usage counts across retrieval strategies on WebShop, estimated via kernel density estimation (KDE) (Chen, 2017). Each panel shows how frequently stored memory instances are accessed under a given strategy.

Table 8: Impact of intrinsic feedback on trajectory diversity on WebShop, measured by the Vendi Score (Friedman & Dieng, 2023). A score of 0.00 for Qwen-2.5-7B-Instruct indicates that fewer than two successful trajectories were generated, precluding diversity measurement.

Method	**Intrinsic Feedback**	**Vendi Score (↑)**	
		Successful Traj.	Failed Traj.
Qwen-2.5-7B-Instruct	–	0.00*	1.89
GRPO (Baseline)	–	1.85	1.71
+ Capability-Evolution Rewards	Numerical	2.04	1.82
+ SimUtil-UCB Memory Retrieval	Language	2.13	1.97
RETROAGENT (In-Context Self-Reflection)	Dual	2.01	1.78
RETROAGENT (RL-Trained Self-Reflection)	Dual	**2.20**	**1.94**

Intrinsic feedback promotes useful exploration. All intrinsic-feedback variants obtain higher Vendi Scores on successful trajectories than GRPO, indicating more diverse successful strategies. The in-context RETROAGENT variant shows slightly lower diversity than the single-feedback ablations, suggesting that the two feedback channels may partially constrain exploration when used jointly (Table 3).



(a) Impact of the relevance–utility tradeoff coefficient α on RETROAGENT (in-context self-reflection).

(b) Impact of the self-reflection objective weight λ_{reflect} on RETROAGENT (RL-trained self-reflection).

Figure 9: Sensitivity of RETROAGENT to two key coefficients on the WebShop test set, reported in terms of task score and success rate (averaged over three runs).

F Sensitivity to the Relevance–Utility Tradeoff Coefficient

We examine the relevance–utility tradeoff in memory retrieval for RetroAgent with in-context self-reflection by varying the coefficient α from 0.3 (utility-oriented) to 0.7 (relevance-oriented). As shown in Figure 9a, RetroAgent achieves higher WebShop task scores and success rates when utility is prioritized ($\alpha = 0.3$). This highlights the importance of accounting for historical utility, rather than relying solely on semantic relevance, when retrieving experiential memories.

G Sensitivity to the Self-Reflection Objective Weight

We examine the effect of the self-reflection objective weight λ_{reflect} on the final performance of RetroAgent with RL-trained self-reflection mechanism by varying λ_{reflect} from 0 (self-reflection loss disabled) to 1. As shown in Figure 9b, increasing λ_{reflect} consistently improves performance on WebShop, raising the success rate from 75.8% to 82.3% and the task score from 86.0% to 88.9%.

H Training Efficiency

We evaluate training efficiency by comparing RetroAgent with the GRPO baseline in terms of training time (Figure 10).

Intrinsic feedback reduces time to target performance. Despite higher total training costs, RetroAgent reaches GRPO's peak performance earlier: the in-context and RL-trained variants match it at steps 65 and 73, respectively (Figure 4a), reducing training time to this level by 46% and 32%. The RL-trained variant's slower progress may stem from jointly optimizing the reflection objective.

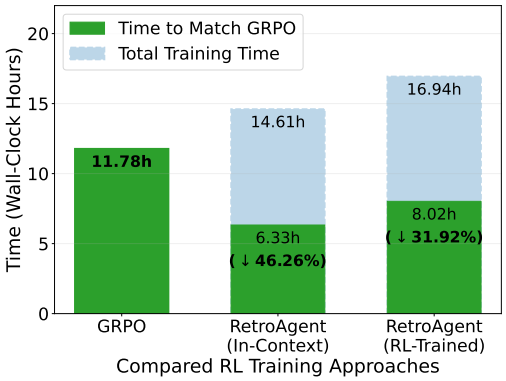


Figure 10: Training time (wall-clock hours) on WebShop. "Time to Match GRPO" denotes the time required for each RetroAgent variant to reach GRPO's peak performance.

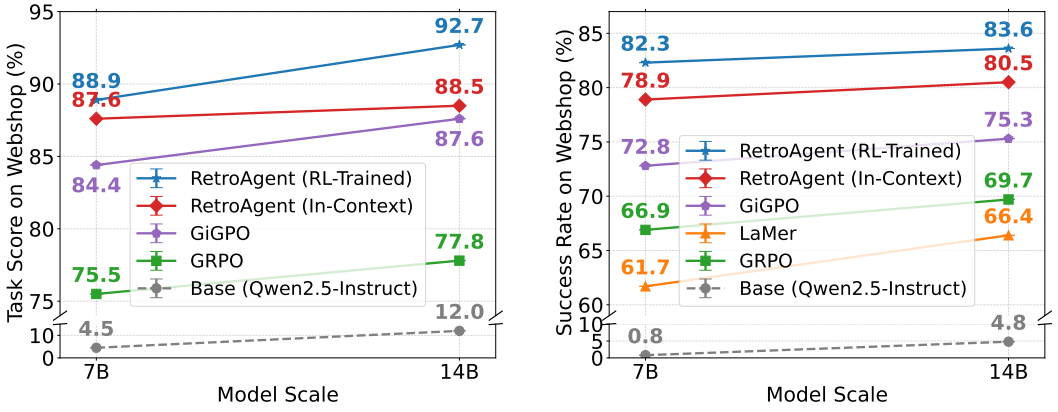
I GENERALIZATION ACROSS MODEL ARCHITECTURES

Table 9: Performance of RETROAGENT on Llama-3.1-8B-Instruct across four agentic benchmarks. All improvements are statistically significant ($p < 0.01$).

Method	ALFWorld	WebShop		Sokoban	MineSweeper
	Success (%)	Score (%)	Success (%)	Success (%)	Success (%)
Llama-3.1-8B-Instruct (Zero-shot)	$29.2_{\pm 0.9}$	$0.2_{\pm 0.4}$	$0.1_{\pm 0.2}$	$5.7_{\pm 0.5}$	$7.0_{\pm 0.8}$
GRPO (Baseline)	$72.7_{\pm 2.3}$	$78.0_{\pm 2.3}$	$67.6_{\pm 2.8}$	$12.2_{\pm 1.2}$	$42.4_{\pm 2.5}$
LAMER (Jiang et al., 2025)	$76.0_{\pm 1.8}$	-	$70.3_{\pm 3.6}$	$15.9_{\pm 2.4}$	$32.0_{\pm 3.4}$
GiGPO (Feng et al., 2025)	$90.9_{\pm 3.6}$	$87.8_{\pm 2.3}$	$77.7_{\pm 3.9}$	$13.5_{\pm 1.2}$	$48.2_{\pm 2.0}$
RETROAGENT (In-Context)	$\mathbf{93.1}_{\pm \mathbf{1.5}}$	$87.8_{\pm 1.8}$	$71.9_{\pm 3.6}$	$\mathbf{39.1}_{\pm \mathbf{1.3}}$	$52.3_{\pm 1.6}$
RETROAGENT (RL-Trained)	$91.4_{\pm 1.4}$	$\mathbf{89.5}_{\pm \mathbf{2.1}}$	$\mathbf{80.5}_{\pm \mathbf{2.2}}$	$24.5_{\pm 2.8}$	$\mathbf{59.9}_{\pm \mathbf{3.2}}$

RETROAGENT generalizes across model architectures. To assess architectural generality, we evaluate RETROAGENT with Llama-3.1-8B-Instruct (Grattafiori et al., 2024). As shown in Table 9, RETROAGENT consistently achieves state-of-the-art performance across all four tasks. The RL-trained self-reflection variant slightly underperforms the in-context variant on ALFWorld and Sokoban, likely due to interference between reflection and decision-making objectives during joint optimization. Specifically, the auxiliary reflection loss may weaken the primary policy-gradient signal. We leave improved multi-objective balancing to future work.

J SCALING ACROSS MODEL SIZES



(a) Task score comparison.

(b) Success rate comparison.

Figure 11: Performance comparison of various methods across different model scales.

RETROAGENT generalizes across model scales. We evaluate RETROAGENT on WebShop using Qwen2.5-Instruct models at 7B and 14B scales. As shown in Figure 11, RETROAGENT consistently outperforms competitive baselines at both scales. Scaling from 7B to 14B yields modest gains: task score improves by $+0.9\%$–$+3.8\%$, and success rate by $+1.3\%$–$+1.6\%$. These results suggest that larger models can better encode and exploit experiential knowledge, but that model scaling alone provides limited benefits when the compared models share similar pretraining distributions and thus similar underlying reasoning capabilities.

K QUALITATIVE ANALYSIS

We qualitatively examine RETROAGENT's policy evolution by analyzing how lessons distilled from similar past tasks guide decision-making during training. We compare an early failed trajectory from RETROAGENT with in-context self-reflection at step 65 and a later successful trajectory at

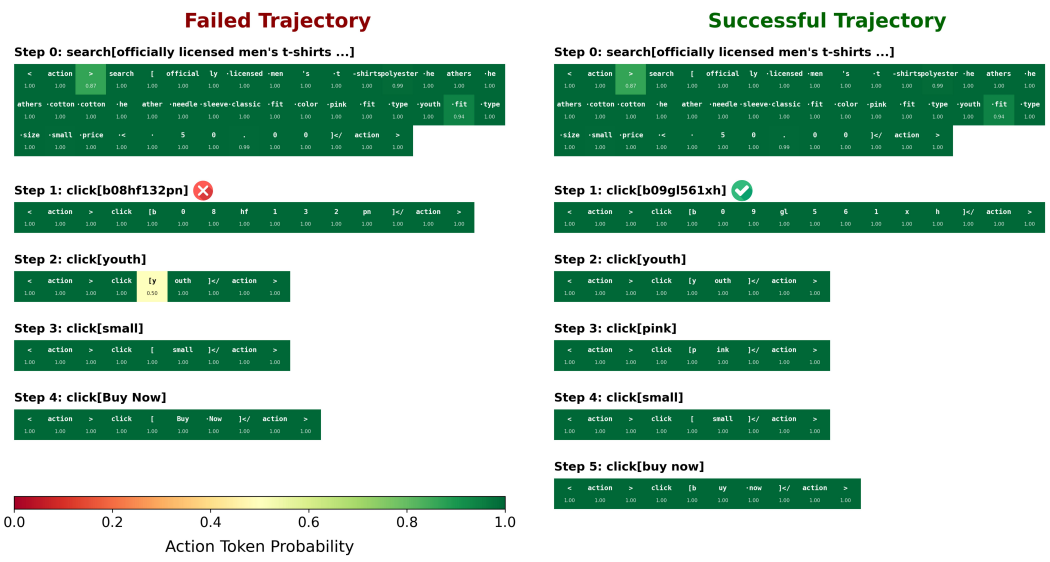


Figure 12: Qualitative comparison of RETROAGENT (in-context self-reflection) on the WebShop validation set between training step 65 (failed trajectory, left) and training step 150 (successful trajectory, right). For conciseness, only action tokens and their corresponding probabilities are shown at each decision step.

step 150 on the WebShop validation set. As shown in Figure 12, at step 65, RETROAGENT selects an incorrect item at decision Step 1, fails to choose the required pink variant, and assigns low token-level confidence to the correct category "youth." In contrast, at step 150, RETROAGENT retrieves relevant lessons from memory and selects the correct item with the required attributes at higher confidence. Full trajectories are provided in Appendix P.

L Extended Related Work

LLMs as Decision-Making Agents. The reasoning capabilities of LLMs have driven their deployment as autonomous decision-making agents. An initial line of research prompts frozen LLMs: ReAct (Yao et al., 2022c), Reflexion (Shinn et al., 2023), and related methods (Park et al., 2023; Wang et al., 2024a) leverage in-context examples, structured prompts, memory retrieval (Wang et al., 2024b), and external tools (Schick et al., 2023; Xie et al., 2024; Zhang et al., 2025a) to tackle complex tasks. However, these approaches are inherently bounded by the capabilities of the underlying foundation model. This ceiling has motivated a second line of work that trains LLM agents directly—through supervised fine-tuning (Tajwar et al., 2025; Xi et al., 2025) or RL (Song et al., 2024; Zhang et al., 2025b; Feng et al., 2025; Jiang et al., 2025)—enabling them to improve from environmental interactions rather than relying on static prompts or handcrafted workflows.

Learning from Experience through Retrospection. A growing body of work moves beyond scalar rewards by leveraging verbal feedback and retrospective memory for agent self-improvement. Early approaches (Shinn et al., 2023; Madaan et al., 2023; Yao et al., 2024) generate natural-language critiques or lessons from interactions, iteratively refining same-task performance via in-context learning. Subsequent work internalizes such feedback into model parameters: Jiang et al. (2025) use reflections to guide cross-episode adaptation within a meta-RL framework, while Zhang et al. (2025c); Hübotter et al. (2026) refine failed trajectories into high-quality data for policy optimization through RL or distillation. A complementary direction adopts memory-based architectures (Goyal et al., 2022; Wu et al., 2025; Wang et al., 2025c; Zhang et al., 2026; Zhou et al., 2025; Fang et al., 2026; Liu et al., 2026b) that store trajectories, lessons, or skills (Xia et al., 2026) in a retrieval buffer to assist similar future tasks in context. RetroAgent advances this paradigm along a new axis: the agent reflects on its trajectories to produce both intrinsic numerical rewards that guide exploration and intrinsic language feedback that facilitates exploiting past experiences, with these dual signals jointly driving policy optimization.

Reward Shaping, Dense Feedback, and Learned Verifiers. Sparse-reward exploration is traditionally addressed via potential-based reward shaping (Ng et al., 1999), heuristic progress signals, or learned process reward models (Cobbe et al., 2021; Uesato et al., 2022; Lightman et al., 2024; Wang et al., 2024c). Unlike potential-based shaping, which guarantees policy invariance through state-difference potentials, practical progress verifiers typically require manual milestone engineering, environment hooks, or dense step-level supervision. RetroAgent departs from these approaches along three axes:

(1) *Autonomous Hindsight Evaluation:* Rather than relying on hand-crafted hooks, our LLM reflector evaluates raw multi-turn histories retrospectively, combining scalar progress estimation with actionable linguistic feedback without external reward-model annotations.

(2) *Capability Evolution vs. Static Credit (Mitigating Reward Gaming):* Static progress shaping induces *reward gaming* (Randløv & Alstrøm, 1998; Amodei et al., 2016), where agents exploit misspecified proxy objectives by repeatedly cycling through easy intermediate milestones without completing the target task. In contrast, RetroAgent tracks a dynamic capability watermark, granting intrinsic credit only when a trajectory exceeds the agent's historical performance on that task.

(3) *Robustness and Invariance (Mitigating Reward Hacking):* While trajectory-level credit lacks the formal policy-invariance guarantee of potential-based shaping (Ng et al., 1999), we actively prevent *reward hacking* and verifier overoptimization (Skalse et al., 2022; Gao et al., 2023)—the phenomenon where a policy exploits verifier blind spots, proxy discrepancies, or hallucinated progress. Specifically: (i) intrinsic bonuses decay as tasks are mastered, ensuring asymptotic convergence is governed solely by extrinsic goals; (ii) group-relative advantage normalization in GRPO (Shao et al., 2024b) restricts intrinsic feedback to tie-breaking among failed rollouts within a batch; and (iii) the reflection module is trained directly against ground-truth terminal success, penalizing ungrounded progress assessments and score inflation.

M THEORETICAL ANALYSIS: SYNERGISTIC BENEFIT OF DUAL FEEDBACK

We analyze how reflection memory and intrinsic numerical feedback jointly affect policy optimization. Memory retrieval changes the rollout distribution by increasing the probability of sampling trajectories with high reflected potential, while the capability-evolution reward assigns larger return to such trajectories. Thus, trajectories exhibiting partial progress can be reinforced even without extrinsic success.

Let $q_0 = x$ be the original prompt and

$$q_m = f_{\text{memory}}(x, \mathcal{M})$$

be the memory-augmented prompt. For a fixed behavior policy $\pi_{\theta_{\text{old}}}$, define

$$Q_0(\tau) = p_{\theta_{\text{old}}}(\tau \mid q_0), \qquad Q_m(\tau) = p_{\theta_{\text{old}}}(\tau \mid q_m).$$

If memory-augmented rollouts are sampled with probability $\rho \in [0, 1]$, the rollout distribution is

$$Q_\rho(\tau) = (1 - \rho)Q_0(\tau) + \rho Q_m(\tau). \tag{2}$$

In RETROAGENT, $\rho = 1/2$.

Recall that τ is δ-valuable at iteration k if

$$\phi_{(x,\tau),k} \geq \Phi_x^{(k)} + \delta.$$

Let

$$p_{k,\delta}^0 = \Pr_{\tau \sim Q_0}\left[\tau \in \mathcal{V}_{k,\delta}(x)\right], \qquad p_{k,\delta}^m = \Pr_{\tau \sim Q_m}\left[\tau \in \mathcal{V}_{k,\delta}(x)\right].$$

Then, under Q_ρ,

$$p_{k,\delta}^\rho = \Pr_{\tau \sim Q_\rho}\left[\tau \in \mathcal{V}_{k,\delta}(x)\right] = (1 - \rho)p_{k,\delta}^0 + \rho p_{k,\delta}^m. \tag{3}$$

Intrinsic feedback as selective gradient weighting. At iteration k, the augmented reward is

$$R_k^{\text{aug}}(\tau) = R^{\text{ext}}(\tau) + R_k^{\text{int}}(\tau), \qquad R_k^{\text{int}}(\tau) = \left[\phi_{(x,\tau),k} - \Phi_x^{(k)}\right]_+. \tag{4}$$

For $q \in \{q_0, q_m\}$, define

$$J_k^{\text{aug}}(\theta; q) = \mathbb{E}_{\tau \sim p_\theta(\cdot|q)}\left[R_k^{\text{aug}}(\tau)\right]. \tag{5}$$

Treating rewards as trajectory-level feedback, the likelihood-ratio identity gives

$$\begin{aligned} \nabla_\theta J_k^{\text{aug}}(\theta; q) &= \mathbb{E}_{\tau \sim p_\theta(\cdot|q)}\left[R_k^{\text{aug}}(\tau)\nabla_\theta \log p_\theta(\tau \mid q)\right] \\ &= \mathbb{E}\left[R^{\text{ext}}(\tau)\nabla_\theta \log p_\theta(\tau \mid q)\right] + \mathbb{E}\left[R_k^{\text{int}}(\tau)\nabla_\theta \log p_\theta(\tau \mid q)\right]. \end{aligned} \tag{6}$$

Thus, intrinsic feedback adds the gradient component

$$g_k^{\text{int}}(\theta; q) = \mathbb{E}_{\tau \sim p_\theta(\cdot|q)}\left[R_k^{\text{int}}(\tau)\nabla_\theta \log p_\theta(\tau \mid q)\right]. \tag{7}$$

For every $\tau \in \mathcal{V}_{k,\delta}(x)$, $R_k^{\text{int}}(\tau) \geq \delta$, whereas trajectories below the current baseline receive no intrinsic reward. Hence, the capability-evolution reward selectively upweights trajectories whose reflected potential exceeds the current capability threshold.

In practice, returns are replaced by advantages. Let

$$A_k^{\text{aug}}(\tau, q) = R_k^{\text{aug}}(\tau) - b_k(q), \tag{8}$$

where $b_k(q)$ is an action-independent baseline, such as the GRPO group mean. Since

$$\mathbb{E}_{\tau \sim p_\theta(\cdot|q)}\left[b_k(q)\nabla_\theta \log p_\theta(\tau \mid q)\right] = 0,$$

the policy gradient is equivalently

$$\nabla_\theta J_k^{\text{aug}}(\theta; q) = \mathbb{E}_{\tau \sim p_\theta(\cdot|q)}\left[A_k^{\text{aug}}(\tau, q)\nabla_\theta \log p_\theta(\tau \mid q)\right]. \tag{9}$$

Therefore, even if $R^{\text{ext}}(\tau) = 0$, a trajectory can receive positive gradient weight whenever its augmented advantage is positive. Intrinsic feedback consequently separates failures with meaningful reflected progress from uninformative failures.

Proposition 1 (Intrinsic feedback lower-bounds valuable-trajectory weight)**.** *Fix $q \in \{q_0, q_m\}$, and define*

$$p_{k,\delta}(\theta; q) = \Pr_{\tau \sim p_\theta(\cdot|q)} \left[\tau \in \mathcal{V}_{k,\delta}(x)\right].$$

For

$$J_k^{\text{int}}(\theta; q) = \mathbb{E}_{\tau \sim p_\theta(\cdot|q)} \left[R_k^{\text{int}}(\tau)\right],$$

we have

$$J_k^{\text{int}}(\theta; q) \geq \delta\, p_{k,\delta}(\theta; q). \tag{10}$$

Moreover,

$$\begin{aligned} \nabla_\theta J_k^{\text{int}}(\theta; q) = \mathbb{E}\left[R_k^{\text{int}}(\tau) \nabla_\theta \log p_\theta(\tau \mid q) \mathbf{1}\{\tau \in \mathcal{V}_{k,\delta}(x)\}\right] \\ + \mathbb{E}\left[R_k^{\text{int}}(\tau) \nabla_\theta \log p_\theta(\tau \mid q) \mathbf{1}\{\tau \notin \mathcal{V}_{k,\delta}(x)\}\right], \end{aligned} \tag{11}$$

where the first term assigns at least δ intrinsic return weight to each δ-valuable trajectory.

Proof. For $\tau \in \mathcal{V}_{k,\delta}(x)$, $R_k^{\text{int}}(\tau) \geq \delta$, and otherwise $R_k^{\text{int}}(\tau) \geq 0$. Hence,

$$\begin{aligned} J_k^{\text{int}}(\theta; q) &= \mathbb{E}\left[R_k^{\text{int}}(\tau)\right] \\ &\geq \mathbb{E}\left[\delta\, \mathbf{1}\{\tau \in \mathcal{V}_{k,\delta}(x)\}\right] \\ &= \delta\, p_{k,\delta}(\theta; q). \end{aligned}$$

The gradient decomposition follows by inserting

$$1 = \mathbf{1}\{\tau \in \mathcal{V}_{k,\delta}(x)\} + \mathbf{1}\{\tau \notin \mathcal{V}_{k,\delta}(x)\}$$

into the likelihood-ratio gradient. □

Joint effect. The mixture objective is

$$J_k^\rho(\theta) = (1-\rho) J_k^{\text{aug}}(\theta; q_0) + \rho J_k^{\text{aug}}(\theta; q_m). \tag{12}$$

Its gradient is

$$\begin{aligned} \nabla_\theta J_k^\rho(\theta) = (1-\rho)\mathbb{E}_{\tau \sim Q_0}\left[A_k^{\text{aug}}(\tau, q_0) \nabla_\theta \log p_\theta(\tau \mid q_0)\right] \\ + \rho \mathbb{E}_{\tau \sim Q_m}\left[A_k^{\text{aug}}(\tau, q_m) \nabla_\theta \log p_\theta(\tau \mid q_m)\right]. \end{aligned} \tag{13}$$

Thus, memory retrieval affects the sampling distribution, while intrinsic feedback affects return and advantage weights.

Theorem 2 (Joint improvement from memory retrieval and intrinsic reward)**.** *Assume memory augmentation increases the probability of sampling a δ-valuable trajectory by at least $\Delta_{\mathcal{M}} > 0$:*

$$p_{k,\delta}^m \geq p_{k,\delta}^0 + \Delta_{\mathcal{M}}. \tag{14}$$

Then

$$p_{k,\delta}^\rho \geq p_{k,\delta}^0 + \rho \Delta_{\mathcal{M}}, \tag{15}$$

and

$$\mathbb{E}_{\tau \sim Q_\rho}\left[R_k^{\text{int}}(\tau)\right] \geq \delta\left(p_{k,\delta}^0 + \rho \Delta_{\mathcal{M}}\right). \tag{16}$$

Therefore, relative to base-only sampling, the intrinsic-reward lower bound improves by at least $\rho\delta\Delta_{\mathcal{M}}$. For $\rho = 1/2$, the gain is at least $\delta\Delta_{\mathcal{M}}/2$.

Proof. By Equation equation 3 and Equation equation 14,

$$\begin{aligned} p_{k,\delta}^\rho &= (1-\rho) p_{k,\delta}^0 + \rho p_{k,\delta}^m \\ &\geq (1-\rho) p_{k,\delta}^0 + \rho\left(p_{k,\delta}^0 + \Delta_{\mathcal{M}}\right) \\ &= p_{k,\delta}^0 + \rho \Delta_{\mathcal{M}}. \end{aligned}$$

Since every δ-valuable trajectory receives intrinsic reward at least δ, and all intrinsic rewards are nonnegative,

$$\mathbb{E}_{\tau \sim Q_\rho}\left[R_k^{\text{int}}(\tau)\right] \geq \delta p_{k,\delta}^\rho \geq \delta\left(p_{k,\delta}^0 + \rho \Delta_{\mathcal{M}}\right).$$

□

Internalization through shared parameters. Because base and memory-augmented prompts share policy parameters, a high-advantage trajectory sampled under one prompt can increase the likelihood of related behavior under the other prompt when their score gradients are aligned.

Let $\tau^+ \in \mathcal{V}_{k,\delta}(x)$ be sampled under $q_a \in \{q_0, q_m\}$, and suppose $A_k^{\text{aug}}(\tau^+, q_a) > 0$. A single-sample update is

$$\theta^+ = \theta + \eta A_k^{\text{aug}}(\tau^+, q_a) \nabla_\theta \log p_\theta(\tau^+ \mid q_a), \tag{17}$$

with learning rate $\eta > 0$. If τ^+ is extrinsically failed but valuable, then $R^{\text{ext}}(\tau^+) = 0$ and $R_k^{\text{int}}(\tau^+) \geq \delta$, so it may still be reinforced whenever its augmented advantage is positive.

Proposition 3 (Internalization of memory-guided exploration)**.** *Let $\tau^+ \in \mathcal{V}_{k,\delta}(x)$ be sampled under $q_a \in \{q_0, q_m\}$, and let $q_b \in \{q_0, q_m\}$. Under Equation equation 17,*

$$\log p_{\theta^+}(\tau^+ \mid q_b) - \log p_\theta(\tau^+ \mid q_b) = \eta A_k^{\text{aug}}(\tau^+, q_a) \left\langle \nabla_\theta \log p_\theta(\tau^+ \mid q_b), \nabla_\theta \log p_\theta(\tau^+ \mid q_a) \right\rangle + O(\eta^2). \tag{18}$$

Thus, if the inner product is positive, then for sufficiently small η the update increases the likelihood of τ^+ under q_b to first order.

Proof. A first-order Taylor expansion gives

$$\log p_{\theta^+}(\tau^+ \mid q_b) = \log p_\theta(\tau^+ \mid q_b) + \left\langle \nabla_\theta \log p_\theta(\tau^+ \mid q_b), \theta^+ - \theta \right\rangle + O(\|\theta^+ - \theta\|_2^2).$$

Substituting Equation equation 17 proves Equation equation 18. Since $A_k^{\text{aug}}(\tau^+, q_a) > 0$, positive alignment gives a positive first-order change. □

Corollary 1 (Expected transfer from memory-augmented to base prompting)**.** *Define*

$$h_0(\tau) = \nabla_\theta \log p_\theta(\tau \mid q_0), \qquad h_m(\tau) = \nabla_\theta \log p_\theta(\tau \mid q_m).$$

Suppose

$$\mathbb{E}_{\tau \sim Q_m} \left[A_k^{\text{aug}}(\tau, q_m) \left\langle h_0(\tau), h_m(\tau) \right\rangle \mathbf{1}\{\tau \in \mathcal{V}_{k,\delta}(x)\} \right] > 0. \tag{19}$$

Then, for sufficiently small η, a policy-gradient update on memory-augmented valuable trajectories increases their expected log-likelihood under the base prompt to first order:

$$\mathbb{E}_{\tau \sim Q_m} \left[(\log p_{\theta^+}(\tau \mid q_0) - \log p_\theta(\tau \mid q_0)) \mathbf{1}\{\tau \in \mathcal{V}_{k,\delta}(x)\} \right] > 0, \tag{20}$$

up to $O(\eta^2)$ terms.

Proof. Apply Proposition 3 with $q_a = q_m$ and $q_b = q_0$, multiply by $\mathbf{1}\{\tau \in \mathcal{V}_{k,\delta}(x)\}$, and take expectation over $\tau \sim Q_m$. The first-order term is positive by Equation equation 19; for sufficiently small η, it dominates the $O(\eta^2)$ remainder. □

Overall, memory retrieval shifts sampling toward trajectories with higher reflected potential, intrinsic numerical feedback upweights trajectories that exceed the current capability baseline, and shared parameters allow high-advantage memory-guided behaviors to transfer to the base prompt when score gradients are aligned. Together, these mechanisms explain how RETROAGENT improves exploration, credit assignment, and online adaptation under sparse extrinsic rewards.

N IMPLEMENTATION DETAILS

Table 10: Default hyperparameters and training configurations for RETROAGENT across all environments.

Hyperparameter	Qwen-2.5-7B-Instruct	Llama-3.1-8B-Instruct	Description
Training Configuration			
Training batch size	16	16	Accumulated batch size per update
Validation batch size	128	128	Batch size for validation
Learning rate	10^{-6}	10^{-6}	Optimizer learning rate
Max prompt length	16 384	16 384	Maximum input context length (tokens)
Max response length	2 048	2 048	Maximum generated response length (tokens)
Group size (N)	8	8	Number of rollouts per prompt
Total steps	150 / 300	150 / 300	Training epochs (150 for ALFWorld and WebShop; 300 for Sokoban and Minesweeper)
Evaluation frequency	5	5	Epochs between consecutive evaluations
Reward and Regularization			
Extrinsic reward (R^{ext})	$\{0,\ 10\}$	$\{0,\ 10\}$	Scalar reward from the environment
Intrinsic reward (R^{int})	$[0,\ 1]$	$[0,\ 1]$	Capability-evolution intrinsic reward
KL coefficient (β)	0.01	0.01	KL-divergence regularization weight
Discount factor (γ)	0.95	0.95	Discount factor for multi-step returns
Memory and Sampling			
Training temperature	0.4	0.4	Sampling temperature during rollouts
Validation temperature	0.4	0.4	Sampling temperature during validation
Initial utility score	0.5	0.5	Initial utility assigned to each memory entry
Utility smoothing (β_{util})	0.05	0.05	Exponential moving average coefficient for utility updates
UCB exploration constant (c)	1.0	1.0	Exploration coefficient in UCB-based retrieval
Relevance–utility weight (α)	0.7	0.7	Trade-off coefficient in retrieval scoring
Memory-augmented ratio	1:1	1:1	Ratio of memory-augmented to base rollouts
Self-Reflection (RL-Trained Variant)			
Reflection reward (R^{reflect})	$\{0,\ 10\}$	$\{0,\ 10\}$	Scalar reward for reflection accuracy
Reflection weight (λ_{reflect})	1.0	1.0	Weight of the self-reflection objective relative to the decision-making objective
Evaluation Configuration			
Evaluation temperature	0.4	0.4	Sampling temperature during evaluation
Max inference tokens	2 048	2 048	Maximum token budget per inference step

Environments. We evaluate RETROAGENT across four distinct agentic tasks: (i) ALFWorld (Shridhar et al., 2021), a text-based embodied environment where agents complete household tasks through navigation and object interaction. We assess both in-distribution (seen rooms) and out-of-distribution (unseen rooms) generalization. (ii) Webshop (Yao et al., 2022a), a simulated e-commerce environment requiring agents to navigate a web interface to purchase products matching user specifications.

(*iii*) Sokoban (Racanière et al., 2017), a planning-heavy puzzle task where agents must push boxes to target locations. Due to the irreversible nature of pushing actions, errors often render puzzles unsolvable. Complexity is governed by board size and box count; we train on 6×6 boards with 2 boxes, following Jiang et al. (2025). (*iv*) MineSweeper (Li et al., 2024), a logic-based puzzle requiring agents to identify mine locations using numerical clues. Difficulty is controlled by board size and mine density; we train on 6×6 boards with 3 mines. We report Success Rate across all tasks, supplemented by Task Score for WebShop.

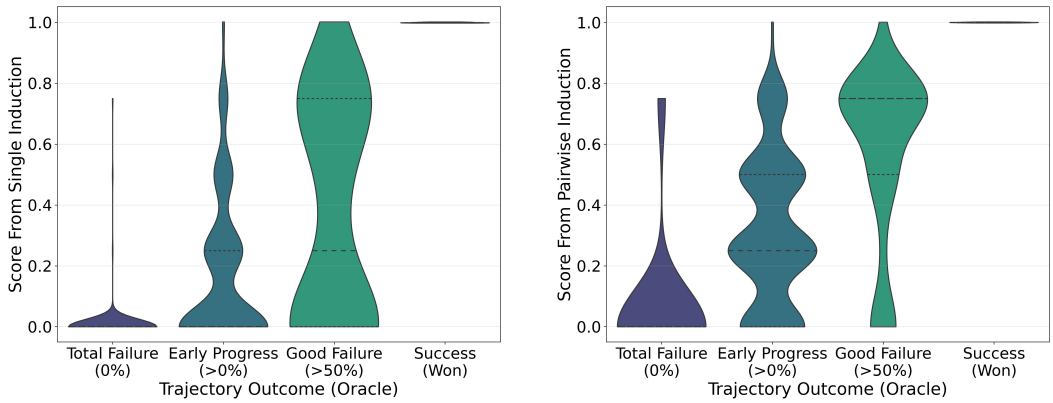
Compared Methods. We evaluate RETROAGENTgent against four categories of competitive baselines, reporting results averaged over three independent runs: (*i*) *Prompting-based methods*: We compare against ReAct (Yao et al., 2022c) and Reflexion (Shinn et al., 2023), the latter of which incorporates an in-context self-reflection mechanism for iterative refinement. (*ii*) *RL algorithms*: We include REINFORCE Leave-One-Out (RLOO) (Kool et al., 2019; Ahmadian et al., 2024), GRPO (Shao et al., 2024b), and Group-in-Group Policy Optimization (GiGPO) (Feng et al., 2025). GiGPO represents the current state-of-the-art by utilizing anchor-state grouping for fine-grained credit assignment. (*iii*) *RL-based frameworks*: This category includes memory-augmented methods such as MemRL (Zhang et al., 2026) (which updates a memory bank while keeping the policy frozen), EvolveR (Wu et al., 2025) (which integrates raw trajectories into optimization), and Mem0 (Chhikara et al., 2025)+GRPO and SimpleMem (Liu et al., 2026a)+GRPO, (which incorporate persistent memory into the training process). We also compare against SkillRL (Xia et al., 2026), a hybrid approach (supervised finetuning and RL) that induces actionable skills via a teacher model to guide the student's policy optimization, and GRPO with EMPG (Wang et al., 2025b), which utilizes entropy-modulated policy gradients for long-horizon optimization. (*iv*) *A Meta-RL framework (Beck et al., 2025)*: We compare against LAMER (Jiang et al., 2025), which leverages a multi-episode structure to foster active exploration and robust adaptation within a meta-learning context.

Implementation Details. We evaluate RETROAGENT on Qwen-2.5-7B-Instruct (Qwen et al., 2025) and Llama-3.1-8B-Instruct (Grattafiori et al., 2024). Although RETROAGENT is generally compatible with various RL algorithms, we adopt GRPO as the default and implement our framework by adapting the open-source Verl training library (Sheng et al., 2024). We employ the task prompts from Feng et al. (2025) to enable decision-making via the *ReAct* format (Yao et al., 2022c), in which the model generates step-by-step reasoning before its corresponding action. At training time, the agent distills lessons as memories from trajectories on the training set; at test time, the agent leverages these memories for task completion on the test set. Detailed hyperparameter settings and training configurations are provided in Table 10. We use `sentence-transformers/all-MiniLM-L6-v2`.[4] All experiments were conducted on 4 × NVIDIA H200 GPUs.

[4] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

O SUPERIORITY OF PAIRWISE INDUCTION OVER SINGLE INDUCTION

We analyze reflection sequences generated during GRPO training augmented with either single-trajectory or pairwise-trajectory induction. Specifically, we sample four sequences per task prompt every 10 training steps, randomly selecting 800 sequences from this pool. Using GPT-4o as an external evaluator, we assess two metrics: the alignment between the agent's induced subtask completion scores and oracle outcomes (induced by GPT-4o), and the quality of the agent's summarized lessons (see Appendix Q.3 for evaluation prompts).



(a) Single induction.

(b) Pairwise induction.

Figure 13: Comparison of intrinsic discrimination capability for Qwen-2.5-7B-Instruct using single *vs.* pairwise induction, measured by subtask completion scores on WebShop.

Figure 13 compares the ability of single- and pairwise-trajectory induction to discriminate trajectory quality within the training-free self-reflection mechanism. For this analysis, we define a successful trajectory as having a completion score of 1. While both methods correctly assign minimal scores to "Total Failure" cases, pairwise induction demonstrates superior discrimination for intermediate outcomes. For example, as shown in Figure 13a, single induction struggles to distinguish "Early Progress" from "Total Failure," resulting in a distribution heavily skewed toward zero for partial successes. In contrast, pairwise induction (Figure 13b) effectively recognizes partial progress, assigning distinctly higher scores to "Early Progress" trajectories and establishing a clear separation from total failures.

P GENERATED TRAJECTORIES BY RETROAGENT

Table 11: Successful trajectory generated by RETROAGENT (in-context self-reflection) at training step 150 on the WebShop validation set.

Instruction: You are an expert autonomous agent operating in the WebShop e-commerce environment. Your task is to: Find me officially licensed, machine wash men's t-shirts with polyester heathers, heathers cotton, cotton heather, needle sleeve, classic fit with color: pink, and fit type: youth, and size: small, and price lower than 50.00 dollars.

Past reflections on similar tasks: Action Insight: The search terms used did not match the available products, leading to no suitable options being found. | Navigation Insight: Efficient use of filters could have potentially narrowed down the search, but the specific dimensions and color were not met in any of the results. Warning: These lessons may be outdated. Use them only if they align with your current observation.

Observation 1: Your current observation is: 'Search'. Your admissible actions of the current situation are: `['search[<your query>]', 'click[search]']` Now it's your turn to take one action for the current step. You should first reason step-by-step about the current situation, then think carefully which admissible action best advances the shopping goal. This reasoning process MUST be enclosed within <think> </think> tags. Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

RETROAGENT: <think> To achieve the goal of finding officially licensed, machine wash men's t-shirts with specific attributes, I need to construct a search query that includes all the required attributes... I will construct the search query to include all these attributes and then execute the search. </think> ***Action:*** `search[officially licensed men's t-shirts polyester heathers heathers cotton cotton heather needle sleeve classic fit color pink fit type youth fit type size small price < 50.00]`

Observation 2: You are now at step 2 and your current observation is: 'Back to Search' [SEP] 'Page 1 (Total results: 50)' [SEP] 'Next >' [SEP] 'B09GL561XH' [SEP] 'Disney Mickey And Friends Happy Birthday Mickey Confetti T-Shirt' [SEP] '$22.99' [SEP] 'B07XPR3R7N' [SEP] 'Marvel Avengers: Endgame Captain America America's Language T-Shirt' [SEP] '$22.99' ... [SEP] 'B09Q8RD8YN' [SEP] 'Bungo Stray Anime Dogs Anime Character, Long Sleeve, Sweatshirt, Hoodie, T shirt' [SEP] '$19.99'.

RETROAGENT: <think> The current page does not list any t-shirts that are officially licensed... The best action at this point is to click on the first product listed to see if it is the correct product... </think> ***Action:*** `click[b09gl561xh]`

Observation 3: You are now at step 3... 'fit type' [SEP] 'men' [SEP] 'women' [SEP] 'youth' [SEP] 'color' [SEP] 'white' ... [SEP] 'pink' [SEP] 'size' [SEP] 'small' ... 'Disney Mickey And Friends Happy Birthday Mickey Confetti T-Shirt' [SEP] 'Price: $22.99'.

RETROAGENT: <think> The current page is not displaying the results for the search query that was previously conducted... The fit type 'youth' is one of the specific fit types mentioned in the search query, so I will click on 'youth' to ensure that the filter is set correctly. </think> ***Action:*** `click[youth]`

Observation 4: You are now at step 4... 'fit type' [SEP] 'men' [SEP] 'women' [SEP] 'youth' [SEP] 'color' ... [SEP] 'pink' [SEP] 'size' [SEP] 'small' ... 'Disney Mickey And Friends Happy Birthday Mickey Confetti T-Shirt' [SEP] 'Price: $22.99'.

RETROAGENT: <think> The current situation shows that the search has narrowed down to a specific product... The next step is to select the color pink to ensure that the color is officially licensed and meets the criteria. </think> ***Action:*** `click[pink]`

Observation 5: You are now at step 5... 'Disney Mickey And Friends Happy Birthday Mickey Confetti T-Shirt' [SEP] 'Price: $22.99' ...

RETROAGENT: <think> The current situation shows that the search has narrowed down to a specific product... with the color being pink. The size is not yet selected... The next step is to select the size small to confirm that it meets the criteria. </think> ***Action:*** `click[small]`

Observation 6: You are now at step 6... 'Disney Mickey And Friends Happy Birthday Mickey Confetti T-Shirt' [SEP] 'Price: $22.99' ...

RETROAGENT: <think> The current situation shows that the search has narrowed down to a specific product... The price is $22.99, which is within the budget... I can proceed with clicking on 'Buy Now' to confirm the selection. </think> ***Action:*** `click[buy now]`

Table 12: Failed trajectory generated by RETROAGENT (in-context self-reflection) at training step 65 on the WebShop validation set.

Instruction: You are an expert autonomous agent operating in the WebShop e-commerce environment. Your task is to: Find me officially licensed, machine wash men's t-shirts with polyester heathers, heathers cotton, cotton heather, needle sleeve, classic fit with color: pink, and fit type: youth, and size: small, and price lower than 50.00 dollars.

Observation 1: Your current observation is: 'Search'. Your admissible actions of the current situation are: `['search[<your query>]', 'click[search]']`. Now it's your turn to take one action for the current step. You should first reason step-by-step about the current situation, then think carefully which admissible action best advances the shopping goal. This reasoning process MUST be enclosed within <think> </think> tags. Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

GRPO: <think> To achieve the goal of finding officially licensed, machine wash men's t-shirts with specific attributes, I need to construct a search query that includes all the required attributes... I will construct the search query to include all these attributes and then execute the search. </think> ***Action:*** `search[officially licensed men's t-shirts polyester heathers heathers cotton cotton heather needle sleeve classic fit color pink fit type youth fit type size small price < 50.00]`

Observation 2: You are now at step 2 and your current observation is: 'Back to Search' [SEP] 'Page 1 (Total results: 50)' [SEP] 'Next >' [SEP] 'B08HF132PN' [SEP] 'Flamingo Pink waterbird costume Gift Premium T-Shirt' [SEP] '$19.99' [SEP] 'B07JVVDJ6L' [SEP] 'Canadian Tuxedo T Shirt - Novelty Denim Tux T-Shirt' [SEP] '$19.99' [SEP] 'B09GL561XH' [SEP] 'Disney Mickey And Friends Happy Birthday Mickey Confetti T-Shirt' [SEP] '$22.99' ... [SEP] 'B07HRFSNL4' [SEP] 'Nautica Men's Solid Crew Neck Short-Sleeve Pocket T-Shirt' [SEP] '$16.05 to $40.98'.

GRPO: <think> The current observation indicates that the search results are being displayed, and the first result is "B08HF132PN" with the title "Flamingo Pink waterbird costume Gift Premium T-Shirt" and a price of $19.99. However, this product does not match the specific criteria we are looking for, as it is a costume t-shirt rather than a men's t-shirt, and it is not officially licensed. Given the current search results, the next logical step is to click on the first result to see if it matches the criteria... The most appropriate action here is to click on the first result, "B08HF132PN", to check if it matches the criteria. </think> ***Action:*** `click[b08hf132pn]`

Observation 3: You are now at step 3 and your current observation is: 'Back to Search' [SEP] '< Prev' [SEP] 'fit type' [SEP] 'men' [SEP] 'women' [SEP] 'youth' [SEP] 'size' [SEP] 'small' ... 'Flamingo Pink waterbird costume Gift Premium T-Shirt' [SEP] 'Price: $19.99' [SEP] 'Rating: N.A.' [SEP] 'Description' [SEP] 'Features' [SEP] 'Reviews' [SEP] 'Buy Now'.

GRPO: <think> 1. The current observation shows that the search results are being displayed... 2. The product does not match the search criteria... 3. The next step is to refine the search results to ensure that the search criteria are met. The most appropriate action here is to click on "youth" to filter the results to show only products that are intended for youth. </think> ***Action:*** `click[youth]`

Observation 4: You are now at step 4... 'fit type' [SEP] 'men' [SEP] 'women' [SEP] 'youth' [SEP] 'size' [SEP] 'small' ... 'Flamingo Pink waterbird costume Gift Premium T-Shirt' [SEP] 'Price: $19.99' ...

GRPO: <think> The current observation shows that the search results are being displayed, and the last action was to click on "youth" fit type... However, the product does not match the criteria we are looking for... Given the current situation, the best action to take is to click on "small" size, as we are looking for a men's t-shirt with specific attributes, and the size is part of the search criteria. </think> ***Action:*** `click[small]`

Observation 5: You are now at step 5... 'Flamingo Pink waterbird costume Gift Premium T-Shirt' [SEP] 'Price: $19.99' ...

GRPO: <think> The current observation indicates that the search results are filtered by fit type, size, and price. The last action was to click on "small" size... However, this product does not match the specific criteria we are looking for... Given the current search results, the best action to take is to click on "Buy Now" for the current product to see if it meets the criteria or if it is a close match. </think> ***Action:*** `click[Buy Now]`

Q TASK PROMPTS

Q.1 PROMPT TEMPLATES FOR IN-CONTEXT SELF-REFLECTION

Prompt for Single Induction on WebShop

You are an expert evaluating a WebShop shopping attempt.

Your task is to: {task_description}

You have just completed an attempt at this shopping task. The task was {success} completed.

Trajectory of the attempt:

{current_trajectory}

<think>

Given the task outcome, analyze the trajectory to understand:

1. What subtasks were attempted? (search, filter, select, purchase)
2. Which subtasks succeeded vs failed based on the observations?
3. What specific actions or decisions led to this outcome?
4. What are the 1-2 most valuable lessons from this attempt?

</think>

Output your evaluation as JSON:

```
{
  "subtasks": [
    {
      "name": "search_product",
      "description": "[describe actual search]",
      "status": "[completed or incomplete]"
    },
    {
      "name": "apply_filters",
      "description": "[describe filters used]",
      "status": "[completed or incomplete]"
    },
    {
      "name": "select_item",
      "description": "[describe selection]",
      "status": "[completed or incomplete]"
    },
    {
      "name": "complete_purchase",
      "description": "[describe purchase]",
      "status": "[completed or incomplete]"
    }
  ],
  "task_success": [true if successfully completed, false if
  unsuccessfully completed],
  "action_lesson": "[key action insight, e.g., 'Precise search with
  brand+model found exact match' OR 'Generic search missed required
  features']",
  "navigation_lesson": "[navigation insight, e.g., 'Efficient use of
  filters saved time' OR 'Failed to check additional pages with better
  options']"
}
```

EVALUATION GUIDELINES:

- The task outcome has been provided - use it to set `task_success` accordingly
- Focus on WHY the attempt had this outcome:
  - If successful: What strategies worked well?
  - If unsuccessful: What went wrong and where?

- Each subtask status must reflect actual trajectory events
- Lessons should explain factors that led to the outcome
- Reference specific elements from trajectory (item IDs, pages, search terms)
- Use `null` for lessons only if truly not applicable

Prompt for Pairwise Induction on WebShop

You are an expert evaluating a WebShop shopping attempt.

Your task is to: {task_description}

You have just completed an attempt at this shopping task. The task was {success} completed.

{reference_trajectory}

Trajectory of the attempt:

{current_trajectory}

<think>

If a reference trajectory exists, compare it with the current trajectory.

Given the task outcome, analyze the trajectory to understand:

1. What subtasks were attempted? (search, filter, select, purchase)
2. Which subtasks succeeded vs failed based on the observations?
3. What specific actions or decisions led to this outcome?
4. What are the 1-2 most valuable lessons from this attempt?

</think>

Output your evaluation as JSON:

```
{
  "subtasks": [
    {
      "name": "search_product",
      "description": "[describe actual search]",
      "status": "[completed or incomplete]"
    },
    {
      "name": "apply_filters",
      "description": "[describe filters used]",
      "status": "[completed or incomplete]"
    },
    {
      "name": "select_item",
      "description": "[describe selection]",
      "status": "[completed or incomplete]"
    },
    {
      "name": "complete_purchase",
      "description": "[describe purchase]",
      "status": "[completed or incomplete]"
    }
  ],
  "task_success": [true if successfully completed, false if
  unsuccessfully completed],
  "action_lesson": "[key action insight, e.g., 'Precise search with
  brand+model found exact match' OR 'Generic search missed required
  features']",
  "navigation_lesson": "[navigation insight, e.g., 'Efficient use of
  filters saved time' OR 'Failed to check additional pages with better
  options']"
}
```

EVALUATION GUIDELINES:

- The task outcome has been provided - use it to set `task_success` accordingly
- Focus on WHY the attempt had this outcome:
  - If successful: What strategies worked well?
  - If unsuccessful: What went wrong and where?
- Each subtask status must reflect actual trajectory events
- Lessons should explain factors that led to the outcome
- Reference specific elements from trajectory (item IDs, pages, search terms)
- Use `null` for lessons only if truly not applicable

Output ONLY the JSON evaluation.

Prompt for Pairwise Induction on ALFWorld

You are an expert evaluating an ALFRED Embodied Environment task attempt.

Your task is to: {task_description}

You have just completed an attempt at this task. The task was {success} completed.

{reference_trajectory}

Trajectory of the attempt:

{current_trajectory}

<think>

If a reference trajectory exists, compare it with the current trajectory.

Given the task outcome, analyze the trajectory to understand:

1. What subtasks were attempted? (pick up, navigate, use appliance, place object)
2. Which subtasks succeeded vs failed based on the observations?
3. What specific actions or decisions led to this outcome?
4. What is the most valuable lesson from this attempt?

</think>

Output your evaluation as JSON:

```
{
  "subtasks": [
    {
      "name": "pick_up_object",
      "description": "[describe pickup action, e.g., 'Pick up mug from
      countertop']",
      "status": "[completed or incomplete]"
    },
    {
      "name": "navigate_to_location",
      "description": "[describe navigation, e.g., 'Go to microwave
      1']",
      "status": "[completed or incomplete]"
    },
    {
      "name": "use_appliance",
      "description": "[describe appliance use, e.g., 'Heat mug in
      microwave']",
      "status": "[completed or incomplete]"
    },
    {
      "name": "place_object",
      "description": "[describe placement, e.g., 'Place heated mug in
      cabinet']",
      "status": "[completed or incomplete]"
    }
  ],
```

```
  "task_success": [true if successfully completed task goal, false if
  failed],
  "action_lesson": "[key action insight, e.g., 'Attempted to place mug
  1 directly in cabinet 2 without heating - must use microwave 1
  first' OR 'Successfully found knife in drawer 3 after checking wrong
  locations']",
  "navigation_lesson": "[spatial insight, e.g., 'Microwave 1 located
  in kitchen area, not near cabinets' OR 'Multiple sinkbasins exist -
  must check all for target object']"
}
```

EVALUATION GUIDELINES:

- The task outcome has been provided - use it to set `task_success` accordingly
- Focus on WHY the attempt had this outcome:
  - If successful: What sequence or strategy worked well?
  - If unsuccessful: What step failed or was missed?
- Each subtask status must reflect actual trajectory events
- Lessons should explain factors that led to the outcome
- Reference specific elements from trajectory (object IDs, locations, appliances)
- Use `null` for lessons only if truly not applicable

Prompt for Pairwise Induction on Minesweeper

You are an expert evaluating a Minesweeper game attempt.

Task Requirements: Reveal all non-mine cells on a {board_size}x{board_size} board with {n_mines} mines without detonating any mine.

You have just completed an attempt at this Minesweeper game. The game was {success} completed.

{reference_trajectory}

Current Trajectory of the attempt:

{current_trajectory}

`<think>`

If a reference trajectory exists, compare it with the current trajectory.

Analyze the current trajectory to determine:

1. Which subtasks were attempted and their completion status
2. Specific actions/decisions that caused the outcome
3. What went wrong (if failed) or right (if succeeded)
4. Devise a concise, new plan of action that accounts for any mistakes with reference to specific actions that should be taken in the next trial

Game notation for reference:

- Cell states: ? (unopened), . (blank/no neighbors), 1-8 (mine count), * (mine)
- Coordinates: rows/columns indexed 1 to {board_size}
- Valid actions: (row, col) where $1 \leq$ row,col $\leq$ {board_size}
- Blank cells auto-cascade to reveal connected blanks + borders

Subtask Completion Criteria (binary evaluation for failed trajectories too):

- `valid_moves`: COMPLETED if made at least 2 valid format moves; INCOMPLETE if mostly invalid actions
- `exploration_progress`: COMPLETED if revealed $> 10\%$ of board; INCOMPLETE if revealed $< 10\%$
- `logical_attempt`: COMPLETED if attempted any deduction (even if wrong); INCOMPLETE if only random/invalid moves
- `error_recovery`: COMPLETED if corrected any error within 3 attempts; INCOMPLETE if repeated same errors

- `cascade_usage`: COMPLETED if triggered or attempted any cascade; INCOMPLETE if only single cell reveals
- `systematic_approach`: COMPLETED if showed any pattern in move selection; INCOMPLETE if purely random

</think>

Required JSON Output:

```
{
  "subtasks": [
    {
      "name": "valid_moves",
      "description": "[e.g., 'Made 5 valid moves like (1,1), (2,3)' or
      'Only invalid formats like (-1,-1)']",
      "status": "[completed/incomplete]"
    },
    {
      "name": "exploration_progress",
      "description": "[e.g., 'Revealed 15 cells (25% of board)' or
      'Only revealed 2 cells']",
      "status": "[completed/incomplete]"
    },
    {
      "name": "logical_attempt",
      "description": "[e.g., 'Tried to use (3,3)=1 constraint' or 'No
      deduction attempts']",
      "status": "[completed/incomplete]"
    },
    {
      "name": "error_recovery",
      "description": "[e.g., 'Fixed format after 2 attempts' or
      'Repeated invalid action 10 times']",
      "status": "[completed/incomplete]"
    },
    {
      "name": "cascade_usage",
      "description": "[e.g., '(1,1) triggered 8-cell cascade' or 'No
      cascade attempts']",
      "status": "[completed/incomplete]"
    },
    {
      "name": "systematic_approach",
      "description": "[e.g., 'Checked corners first' or 'Random
      clicking']",
      "status": "[completed/incomplete]"
    }
  ],
  "trajectory_value": [count of completed subtasks out of 6],
  "task_success": [true if successfully completed, false if
  unsuccessfully completed],
  "next_priority": "[Most important fix, e.g., 'Use valid (row,col)
  format' or 'When cell shows 1, count unopened neighbors']"
}
```

Evaluation Rules:

- Award COMPLETED for ANY positive demonstration, even in failed games
- `valid_moves`: Just need 2+ correctly formatted moves anywhere in trajectory
- `exploration_progress`: 10% is roughly 6 cells on 8x8 board - achievable even if hit mine
- `logical_attempt`: Credit for trying logic, even if conclusion was wrong
- `error_recovery`: Credit for any correction, even if made new errors later
- `cascade_usage`: Credit for choosing corners/edges that could cascade
- `systematic_approach`: Credit for any non-random pattern in moves

- `trajectory_value` helps distinguish quality among failed attempts (0-6 scale)

Output JSON only.

Prompt for Pairwise Induction on Sokoban

You are an expert evaluating a Sokoban game attempt.

Task Requirements: Push all boxes ('X') onto target spots ('O') in the grid without getting them stuck against walls ('#') or in corners.

You have just completed an attempt at this Sokoban level. The game was {success} completed.

{reference_trajectory}

Current Trajectory of the attempt:

{current_trajectory}

`<think>`

If a reference trajectory exists, compare it with the current trajectory.

Given the task outcome, analyze the trajectory to understand:

1. Which subtasks were attempted and their completion status
2. Specific actions/decisions that caused the outcome
3. What went wrong (if failed) or right (if succeeded)
4. Devise a concise, new plan of action that accounts for any mistakes with reference to specific actions that should be taken in the next trial

Game notation for reference:

- Symbols: # (wall), _ (floor), O (target), X (box), P (player), $\surd$ (box on target)
- Coordinates: (row, col)
- Valid actions: ["up", "down", "left", "right"]
- Rules: Push only (no pull), one box at a time, walls block movement.

Subtask Completion Criteria (binary evaluation for failed trajectories too):

- `valid_moves`: COMPLETED if made at least 2 valid directional moves; INCOMPLETE if mostly invalid formats/hallucinations
- `navigation_logic`: COMPLETED if player successfully navigated to a box; INCOMPLETE if stuck hitting walls/looping
- `box_interaction`: COMPLETED if at least one box was pushed to a new coordinate; INCOMPLETE if no boxes moved
- `deadlock_avoidance`: COMPLETED if avoided pushing boxes into unrecoverable corners/walls; INCOMPLETE if immediate deadlock created
- `goal_progress`: COMPLETED if at least one box was placed on a target; INCOMPLETE if 0 boxes on targets
- `systematic_approach`: COMPLETED if moves showed clear intent (e.g., moving behind a box to push); INCOMPLETE if random walking

`</think>`

Required JSON Output:

```
{
  "subtasks": [
    {
      "name": "valid_moves",
      "description": "[e.g., 'Outputted valid directions like up,
      down' or 'Used invalid commands']",
      "status": "[completed/incomplete]"
    },
    {
      "name": "navigation_logic",
```

```
      "description": "[e.g., 'Reached box at (3,2)' or 'Walked into
      wall at (1,1) repeatedly']",
      "status": "[completed/incomplete]"
    },
    {
      "name": "box_interaction",
      "description": "[e.g., 'Pushed box from (2,2) to (2,3)' or 'No
      boxes moved']",
      "status": "[completed/incomplete]"
    },
    {
      "name": "deadlock_avoidance",
      "description": "[e.g., 'Kept boxes away from corners' or 'Pushed
      box into corner (1,1)']",
      "status": "[completed/incomplete]"
    },
    {
      "name": "goal_progress",
      "description": "[e.g., '1/3 boxes placed on target' or 'No boxes
      on targets']",
      "status": "[completed/incomplete]"
    },
    {
      "name": "systematic_approach",
      "description": "[e.g., 'Cleared path for second box' or 'Random
      movement']",
      "status": "[completed/incomplete]"
    }
  ],
  "trajectory_value": [count of completed subtasks out of 6],
  "task_success": [true if successfully completed, false if
  unsuccessfully completed],
  "next_priority": "[Most important fix, e.g., 'Don't push box into
  corner at (1,1)' or 'Move to (2,3) to push down']"
}
```

Evaluation Rules:

- Award COMPLETED for ANY positive demonstration, even in failed games
- `valid_moves`: Just need 2+ correctly formatted actions
- `navigation_logic`: Credit for traversing the map without getting stuck on walls immediately
- `box_interaction`: Credit for changing the state of the board (moving a box)
- `deadlock_avoidance`: Credit if the first box move didn't result in an immediate game-over state
- `goal_progress`: Credit for securing at least one objective, even if others failed
- `systematic_approach`: Credit for positioning the player specifically to push a box
- `trajectory_value` helps distinguish quality among failed attempts (0-6 scale)

Output JSON only.

Q.2 Prompt Templates for RL-Trained Self-Reflection

Prompt for Pairwise Induction on WebShop

You are an expert evaluating a WebShop shopping attempt.

Target Task: {task_description}

You have just completed an attempt at this shopping task.

Trajectory of the attempt:

{current_trajectory}

`<think>`

If a reference trajectory exists, compare it with the current trajectory.

Analyze the trajectory to determine if the task was successful:

1. Identify the specific requirements in the 'Target Task' (attributes, type, options).
2. Examine the final action in the trajectory. Did it end in a 'click[buy]'?
3. If a purchase was made, compare the purchased item's details against the 'Target Task' requirements.
4. Did the purchased item match ALL requirements? (If no purchase was made, it is a failure).
5. What specific actions or decisions led to this outcome?
6. What are the 1-2 most valuable lessons from this attempt?

</think>

Output your evaluation as JSON:

```
{
  "subtasks": [
    {
      "name": "search_product",
      "description": "[describe actual search]",
      "status": "[completed or incomplete]"
    },
    {
      "name": "apply_filters",
      "description": "[describe filters used]",
      "status": "[completed or incomplete]"
    },
    {
      "name": "select_item",
      "description": "[describe selection]",
      "status": "[completed or incomplete]"
    },
    {
      "name": "complete_purchase",
      "description": "[describe purchase]",
      "status": "[completed or incomplete]"
    }
  ],
  "task_success": [true if the correct item was purchased, false
  otherwise],
  "action_lesson": "[key action insight, e.g., 'Precise search with
  brand+model found exact match' OR 'Generic search missed required
  features']",
  "navigation_lesson": "[navigation insight, e.g., 'Efficient use of
  filters saved time' OR 'Failed to check additional pages with better
  options']"
}
```

EVALUATION GUIDELINES:

- **Determine Success Yourself:** You must judge 'task_success' by comparing the purchased item in the trajectory to the Target Task.
- **Criteria for Success:** The task is ONLY true if the agent successfully clicked 'buy' on an item that matches all required attributes (color, size, flavor, etc.).
- **Criteria for Failure:** If the trajectory ends without a purchase, or if the wrong item was bought, 'task_success' is false.
- Each subtask status must reflect actual trajectory events.
- Lessons should explain factors that led to the outcome.
- Reference specific elements from trajectory (item IDs, pages, search terms).
- Use `null` for lessons only if truly not applicable.

Prompt for Pairwise Induction on ALFWorld

You are an expert evaluating an ALFWorld embodied agent attempt.

Target Task: {task_description}

You have just completed an attempt at this household task.

Trajectory of the attempt:

{current_trajectory}

<think>

1. If a reference trajectory exists, compare it with the current trajectory.
2. Analyze the trajectory to determine if the task was successful:
   (a) Identify the specific requirements in the 'Target Task' (target object, required state change, final destination).
   (b) Examine the sequence of actions. Did the agent successfully locate the correct object?
   (c) If a state change was required (clean, heat, cool, slice), was the correct appliance or tool used?
   (d) Did the agent place the object in the correct final receptacle?
   (e) Did the trajectory end with the 'stop' action after achieving the goal state? (If the agent stopped prematurely or failed to stop, it is a failure).
   (f) What specific actions or decisions led to this outcome?
   (g) What are the 1-2 most valuable lessons from this attempt?

</think>

Output your evaluation as JSON:

```
{{
  "subtasks": [
    {{
      "name": "locate_object",
      "description": "[describe search for target object]",
      "status": "[completed or incomplete]"
    }},
    {{
      "name": "acquire_object",
      "description": "[describe picking up target]",
      "status": "[completed or incomplete]"
    }},
    {{
      "name": "modify_state",
      "description": "[describe heating/cleaning/cooling/slicing if
      applicable, else 'N/A']",
      "status": "[completed, incomplete, or N/A]"
    }},
    {{
      "name": "place_object",
      "description": "[describe final placement]",
      "status": "[completed or incomplete]"
    }}
  ],
  "task_success": [true if the goal state was achieved and 'stop' was
  called, false otherwise],
  "action_lesson": "[key action insight, e.g., 'Used microwave to heat
  apple instead of fridge' OR 'Failed to slice bread before
  plating']",
  "navigation_lesson": "[spatial/search insight, e.g., 'systematically
  checked all cabinet receptacles' OR 'wasted steps revisiting empty
  drawers']"
}}
```

EVALUATION GUIDELINES:

- **Determine Success Yourself:** You must judge 'task_success' by comparing the final state in the trajectory to the Target Task.

- **Criteria for Success:** The task is ONLY true if the agent manipulated the correct object, achieved the correct state (e.g., hot, clean), placed it in the correct target, and issued the 'stop' command.
- **Criteria for Failure:** If the trajectory ends without the 'stop' command, or if the agent stopped without completing the goal (e.g., holding the object instead of placing it), 'task_success' is false.
- Each subtask status must reflect actual trajectory events.
- Lessons should explain factors that led to the outcome.
- Reference specific elements from trajectory (object IDs like 'apple 1', receptacle IDs like 'countertop 2').
- Use `null` for lessons only if truly not applicable.

Output ONLY the JSON evaluation.

Prompt for Pairwise Induction on Sokoban

You are an expert evaluating a Sokoban game attempt.

Task Requirements: Push all boxes ('X') onto target spots ('O') in the grid without getting them stuck against walls ('#') or in corners.

You have just completed an attempt at this Sokoban level.

Current Trajectory of the attempt:

{current_trajectory}

<think>

1. If a reference trajectory exists, compare it with the current trajectory.
2. Analyze the trajectory to determine if the task was successful:
   (a) Identify the grid layout and target locations in the 'Target Task'.
   (b) Examine the final board state in the trajectory. Are ALL boxes ('X') placed on targets ('O') resulting in '$\surd$'?
   (c) If the game ended without success, check for deadlocks (boxes stuck in corners or against walls).
   (d) Did the player successfully navigate the player ('P') to push positions without hitting walls repeatedly?
   (e) What specific logic or movement behavior led to this outcome?
   (f) What are the 1-2 most valuable lessons from this attempt?
   (g) Devise a concise, new plan of action that accounts for any mistakes with reference to specific actions that should be taken in the next trial

Game notation for reference:

- Symbols: # (wall), _ (floor), O (target), X (box), P (player), $\surd$ (box on target)
- Coordinates: (row, col)
- Valid actions: ["up", "down", "left", "right"]
- Rules: Push only (no pull), one box at a time, walls block movement.

Subtask Completion Criteria (binary evaluation for failed trajectories too):

- **valid_moves:** COMPLETED if made at least 2 valid directional moves; INCOMPLETE if mostly invalid formats/hallucinations
- **navigation_logic:** COMPLETED if player successfully navigated to a box; INCOMPLETE if stuck hitting walls/looping
- **box_interaction:** COMPLETED if at least one box was pushed to a new coordinate; INCOMPLETE if no boxes moved
- **deadlock_avoidance:** COMPLETED if avoided pushing boxes into unrecoverable corners/walls; INCOMPLETE if immediate deadlock created
- **goal_progress:** COMPLETED if at least one box was placed on a target; INCOMPLETE if 0 boxes on targets
- **systematic_approach:** COMPLETED if moves showed clear intent (e.g., moving behind a box to push); INCOMPLETE if random walking

</think>

Required JSON Output:

```
{{
  "subtasks": [
    {{
      "name": "valid_moves",
      "description": "[e.g., 'Outputted valid directions like up,
      down' or 'Used invalid commands']",
      "status": "[completed/incomplete]"
    }},
    {{
      "name": "navigation_logic",
      "description": "[e.g., 'Reached box at (3,2)' or 'Walked into
      wall at (1,1) repeatedly']",
      "status": "[completed/incomplete]"
    }},
    {{
      "name": "box_interaction",
      "description": "[e.g., 'Pushed box from (2,2) to (2,3)' or 'No
      boxes moved']",
      "status": "[completed/incomplete]"
    }},
    {{
      "name": "deadlock_avoidance",
      "description": "[e.g., 'Kept boxes away from corners' or 'Pushed
      box into corner (1,1)']",
      "status": "[completed/incomplete]"
    }},
    {{
      "name": "goal_progress",
      "description": "[e.g., '1/3 boxes placed on target' or 'No boxes
      on targets']",
      "status": "[completed/incomplete]"
    }},
    {{
      "name": "systematic_approach",
      "description": "[e.g., 'Cleared path for second box' or 'Random
      movement']",
      "status": "[completed/incomplete]"
    }}
  ],
  "trajectory_value": [count of completed subtasks out of 6],
  "task_success": [true if successfully placed all boxes on targets,
  false if deadlock or incomplete],
  "next_priority": "[Most important fix, e.g., 'Don't push box into
  corner at (1,1)' or 'Move to (2,3) to push down']"
}}
```

Evaluation Rules:

- **Determine Success Yourself:** You must judge 'task_success' by comparing the final board state in the trajectory to the Target Task.
- **Criteria for Success:** The task is ONLY true if ALL boxes are on target spots ('$\surd$').
- **Criteria for Failure:** If the trajectory ends with a deadlock, or if the agent stopped before placing all boxes, 'task_success' is false.
- Each subtask status must reflect actual trajectory events.
- Lessons should explain factors that led to the outcome (planning vs. random).
- Reference specific elements from trajectory (coordinates, symbols).
- Use `null` for lessons only if truly not applicable.

Output ONLY the JSON evaluation.

Prompt for Pairwise Induction on Minesweeper

You are an expert evaluating a Minesweeper game attempt.

Task Requirements: Reveal all non-mine cells on a {board_size}x{board_size} board with {n_mines} mines without detonating any mine.

You have just completed an attempt at this Minesweeper game.

Current Trajectory of the attempt:

{current_trajectory}

<think>

1. If a reference trajectory exists, compare it with the current trajectory.
2. Analyze the trajectory to determine if the task was successful:
   (a) Identify the board constraints (size, mine count) in the 'Target Task'.
   (b) Examine the final action in the trajectory. Did it result in a mine detonation (loss) or a cleared board (win)?
   (c) If the game ended without a mine detonation, check if ALL safe cells were revealed.
   (d) Did the player successfully flag mines (optional but helpful) and reveal all safe spots? (If a mine was hit or safe cells remain hidden, it is a failure).
   (e) What specific logic or guessing behavior led to this outcome?
   (f) What are the 1-2 most valuable lessons from this attempt?
   (g) Devise a concise, new plan of action that accounts for any mistakes with reference to specific actions that should be taken in the next trial

Game notation for reference:

- Cell states: ? (unopened), . (blank/no neighbors), 1-8 (mine count), * (mine)
- Coordinates: rows/columns indexed 1 to {board_size}
- Valid actions: (row, col) where $1 \leq$ row,col $\leq$ {board_size}
- Blank cells auto-cascade to reveal connected blanks + borders

Subtask Completion Criteria (binary evaluation for failed trajectories too):

- **valid_moves:** COMPLETED if made at least 2 valid format moves; INCOMPLETE if mostly invalid actions
- **exploration_progress:** COMPLETED if revealed $> 10\%$ of board; INCOMPLETE if revealed $< 10\%$
- **logical_attempt:** COMPLETED if attempted any deduction (even if wrong); INCOMPLETE if only random/invalid moves
- **error_recovery:** COMPLETED if corrected any error within 3 attempts; INCOMPLETE if repeated same errors
- **cascade_usage:** COMPLETED if triggered or attempted any cascade; INCOMPLETE if only single cell reveals
- **systematic_approach:** COMPLETED if showed any pattern in move selection; INCOMPLETE if purely random

</think>

Required JSON Output:

```
{{
  "subtasks": [
    {{
      "name": "valid_moves",
      "description": "[e.g., 'Made 5 valid moves like (1,1), (2,3)' or
      'Only invalid formats like (-1,-1)']",
      "status": "[completed/incomplete]"
    }},
    {{
      "name": "exploration_progress",
      "description": "[e.g., 'Revealed 15 cells (25% of board)' or
      'Only revealed 2 cells']",
      "status": "[completed/incomplete]"
    }},
```

```
    {{
      "name": "logical_attempt",
      "description": "[e.g., 'Tried to use (3,3)=1 constraint' or 'No
      deduction attempts']",
      "status": "[completed/incomplete]"
    }},
    {{
      "name": "error_recovery",
      "description": "[e.g., 'Fixed format after 2 attempts' or
      'Repeated invalid action 10 times']",
      "status": "[completed/incomplete]"
    }},
    {{
      "name": "cascade_usage",
      "description": "[e.g., '(1,1) triggered 8-cell cascade' or 'No
      cascade attempts']",
      "status": "[completed/incomplete]"
    }},
    {{
      "name": "systematic_approach",
      "description": "[e.g., 'Checked corners first' or 'Random
      clicking']",
      "status": "[completed/incomplete]"
    }}
  ],
  "trajectory_value": [count of completed subtasks out of 6],
  "task_success": [true if successfully cleared all safe cells, false
  if detonated mine or incomplete],
  "next_priority": "[Most important fix, e.g., 'Use valid (row,col)
  format' or 'When cell shows 1, count unopened neighbors']"
}}
```

Evaluation Rules:

- **Determine Success Yourself:** You must judge 'task_success' by comparing the final board state in the trajectory to the Target Task.
- **Criteria for Success:** The task is ONLY true if the agent successfully revealed ALL safe cells without detonating a mine.
- **Criteria for Failure:** If the trajectory ends with a mine detonation, or if the agent stopped before revealing all safe cells, 'task_success' is false.
- Each subtask status must reflect actual trajectory events.
- Lessons should explain factors that led to the outcome (logic vs. guessing).
- Reference specific elements from trajectory (coordinates, cell values).
- Use `null` for lessons only if truly not applicable.

Output ONLY the JSON evaluation.

Q.3 Prompts for Analyzing the Quality of Intrinsic Feedback

To assess the fidelity of the intrinsic feedback generated via self-reflection, we employ GPT-4o (OpenAI et al., 2024) as an external judge. Our evaluation focuses on two key components: the accuracy of the induced subtask completion scores (intrinsic rewards) and the quality of the summarized lessons (intrinsic feedback).

To verify the accuracy of the subtask completion scores, we utilize the prompt detailed in Section Q.1. To evaluate the quality of the summarized lessons derived from the agent's trajectories, we use the prompt presented below.

Prompt for Evaluating Summarized Lessons

System Prompt:

You are an expert evaluator of AI Memory Systems. Your goal is to determine the 'Information Gain' and 'Cruciality' of lessons generated by an agent. You must distinguish between generic fluff (low quality) and specific, actionable insights (high quality).

User Prompt:

 # Context

The agent performed a task in a web environment.

Actual Outcome: `{actual_outcome}`

 # Trajectory (History of Actions)

`{trajectory}`

 # Agent's Generated Reflection (containing Lessons)

`{reflection}`

 # Evaluation Task

Analyze the `action_lesson` and `navigation_lesson` in the reflection above.

1. **Specificity**: Is the lesson specific to the UI elements/errors encountered? (e.g., "Clicking 'Submit' failed because the form was empty" vs. "I failed to click").
2. **Causal Accuracy**: Does the lesson correctly identify the root cause of the `{actual_outcome}`?
3. **Utility**: If the agent retrieves this lesson in a future attempt, will it significantly improve the success rate?

 # Output Format (JSON Only)

```
{
  "lesson_quality_score": <int 1-10>,
  "specificity_rating": <"High"|"Medium"|"Low">,
  "utility_rating": <"High"|"Medium"|"Low">,
  "reasoning": "<Short explanation of why this lesson is
  useful/useless>",
  "is_hallucination": <bool, true if lesson mentions events not in
  trajectory>
}
```