Role-Agent: Bootstrapping LLM Agents via Dual-Role Evolution

Xucong Wang[1,2*] **Ziyu Ma**[2] **Shidong Yang**[2] **Tongwen Huang**[2]
Pengkun Wang[1†] **Yong Wang**[2†] **Xiangxiang Chu**[2]
[1]University of Science and Technology of China [2]AMAP, Alibaba Group
GitHub: https://github.com/xuc865/roleagent

Abstract

Although Large Language Model (LLM) agents have demonstrated strong performance on complex tasks, their learning is often limited by inefficient interaction feedback and static training environments, which hinder broader generalization. To address these limitations, this paper introduces Role-Agent, a framework that harnesses a single LLM to function concurrently as both the agent and the environment, enabling a bootstrapped co-evolution. Role-Agent comprises two synergistic components: World-In-Agent (WIA) and Agent-In-World (AIW). In WIA, the LLM acts as the agent and predicts future states after each action; the alignment between predicted and actual states is then used as a process reward, encouraging environment-aware reasoning. In AIW, the LLM analyzes failure modes from failed trajectories and retrieves tasks with similar failure patterns, thereby reshaping the training data distribution for targeted practice. Experiments on multiple benchmarks show that Role-Agent consistently improves performance, yielding an average gain of over 4% over strong baselines.

1 Introduction

Beyond simple question answering, Large Language Model (LLM) agents (Team et al., 2023; Yang et al., 2025; Chen et al., 2025a; Ou et al., 2025; Ma et al., 2026) have found wide application in complex real-world challenges, owing to their unique abilities to think, reason, and reflect (Yao et al., 2022b; Shinn et al., 2023; Liu et al., 2023; Dong et al., 2025a) within their environments. In more dynamic applications such as coding (Jiang et al., 2025), navigation (Comanici et al., 2025), deep research (Citron, 2024; Team et al., 2025), and embodied applications (Shridhar et al., 2020), the multi-turn tool-use and long-horizon capabilities of agents are critical and have therefore been widely explored (Liu et al., 2023; Dong et al., 2025b).

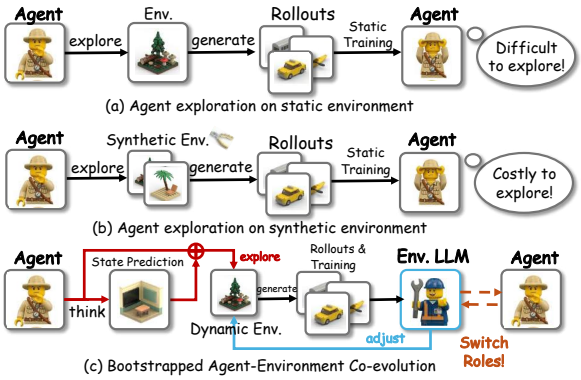

Figure 1: **(a):** Static environments provide sparse and non-specific feedback that limits the agent's exploration; **(b):** Synthetic environments incur high labor and runtime costs; **(c):** The proposed Role-Agent enables one model to switch roles between agent and environment to achieve bootstrapped co-evolution.

Building on the use of Reinforcement Learning (RL) in LLM post-training (Schulman et al., 2017; Rafailov et al., 2023; Chu et al., 2026), Agentic Reinforcement Learning (ARL) incorporates full interaction rollout trajectories into the RL framework, enabling agents to optimize their problem-solving abilities through environment feedback. In contrast to supervised fine-tuning (Zhang and Zhang, 2024), where agents are trained to mimic expert behavior, ARL allows greater solution diversity (Guo et al., 2025) and can substantially enhance agents' reasoning and problem-solving capabilities.

Beyond using expert trajectories and static rewards to optimize agent policies, recent studies of self-evolving agents (Gao et al., 2025) focus on continuous capability growth by autonomously discovering their own deficiencies and updating agent harness (Fernando et al., 2023; Hemberg et al., 2024; Zhang et al., 2025a, 2026b; Anthropic, 2024; Xia et al., 2026). However, most existing methods evolve only the agent itself while treating the environment as a fixed source of tasks, observa-

*Work done during internship at AMAP, Alibaba.
†Project lead: Yong Wang; Corresponding authors: Yong Wang and Pengkun Wang

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 4360–4379
October 24-29, 2026 ©2026 Association for Computational Linguistics

tions, and rewards; The environment fails to expose the agent's hidden weaknesses or provide feedback targeted to its current failure modes. A more desirable paradigm is the synthetic environment, where the agent improves through interaction while the environment also adapts to diagnose the agent's deficiencies and present more challenges. Yet building such an adaptive environment often requires additional environment models, task generators, or scheduling mechanisms (Zhuo et al., 2025; Xue et al., 2026), which increases deployment complexity. This raises a natural question: *can we achieve agent-environment co-evolution by using a single LLM to act as both the agent and the environment?*

Guided by this idea, we propose Role-Agent, which enables bootstrapped agent-environment co-evolution by using a single LLM as a dual-role entity. Role-Agent consists of:

(a) World-In-Agent (WIA), where the LLM agent predicts the future observations resulting from its actions, thereby incorporating environment priors into its rollouts. Role-Agent measures the gap between agent-predicted future states and actual states to estimate the agent's ability to predict action consequences. By integrating this measure into reward and credit assignment, WIA encourages more reliable decision-making in states where action consequences are uncertain.

(b) Agent-In-World (AIW), where the same LLM provides environment feedback and adapts the data distribution to prioritize difficult and easily overlooked tasks. Specifically, we instruct the LLM to analyze failed trajectories step by step, producing failure modes and reflections that reveal the root causes of failure. We then retrieve tasks with similar failure modes and adjust the data distribution accordingly, enabling the agent to focus training on its historical deficiencies.

Extensive experiments demonstrate that Role-Agent consistently outperforms existing approaches, showing that a single LLM can serve as both agent and environment to achieve practical gains in text-based interactive environments. Our contributions are threefold:

- Different from agent-side self-improvement and state-grouped RL methods, we investigate bootstrapped agent-environment co-evolution without human supervision.
- We propose Role-Agent, which uses the World-In-Agent and Agent-In-World modules to cast a single LLM in dual roles, enabling fine-grained environment prediction and adaptive task redistribution.
- Extensive experiments demonstrate that Role-Agent achieves substantial improvements over strong baselines across diverse benchmarks.

2 Related Work

Large Language Model (LLM) Agents. Large language models (LLMs) are increasingly being adopted as autonomous agents across a wide range of domains (Wang et al., 2023, 2024; Jiang et al., 2025; Ou et al., 2025). Early LLM agents were equipped with tool-use (Yao et al., 2022b), reflection (Shinn et al., 2023), or memory schemes (Xu et al., 2025; Fang et al., 2025; Zhang et al., 2026a) to transform LLM backbones into autonomous, interactive agents. More recent studies incorporate RL methods (Lambert et al., 2024) to endow agents with long-horizon reasoning and multi-turn interaction abilities, exemplified by PPO (Schulman et al., 2017), DPO (Rafailov et al., 2023), GRPO (Guo et al., 2025), DAPO (Yu et al., 2025), GSPO (Zheng et al., 2025), and GPG (Chu et al., 2026). While these approaches sample full tool-use trajectories and leverage final outcome rewards with limited extra supervision, another line of studies adopts process reward models (Shao et al., 2024; Zhang et al., 2025b; Wang et al., 2025b) to assign credit to each action, improving complex reasoning tasks.

Self-Evolving Agents. Unlike optimized under fixed data distributions and tasks, self-evolving agents (Gao et al., 2025; Zhai et al., 2025) emphasize autonomous capability iteration within dynamically evolving open environments. EvolveR (Wu et al., 2025) introduces a self-contained lifecycle where the agent distills its own experiences into principles and evolves its policy. Other works (Hu et al., 2024; Novikov et al., 2025) focus on automated exploration of agent design. MAE (Chen et al., 2025b) instantiates three roles (Proposer, Solver and Judge) to co-evolve without human-curated data. More recently, Agentevolver (Zhai et al., 2025) leverages self-questioning, self-navigation, and self-attribution to facilitate agent evolution. GiGPO (Feng et al., 2025) further introduces state-grouped advantage estimation for LLM agent RL. In contrast, Role-Agent achieves bootstrapped agent-environment co-evolution, differing from these studies whose auxiliary roles mainly remain on the agent side.

3 Methodology

3.1 Preliminaries

Problem Setup. We first formalize general multi-step agent-environment interaction tasks as follows: given a task prompt $\boldsymbol{x} \in \mathcal{X}$, the agent generates an action $\boldsymbol{a}_t \in \mathcal{A}$ based on the current state $\boldsymbol{s}_t$ and its policy $\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t, \boldsymbol{x})$ at each step t ($1 \le t \le T$, where T is the interaction length of the trajectory and θ denotes the policy parameters). The environment then provides the next state $\boldsymbol{s}_{t+1}$ and an instant reward r_t. This yields a trajectory (rollout) $\boldsymbol{\tau} = \{(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t)\}_{t=1}^{T}$. We denote a batch of rollouts as $\mathcal{T} = \{\boldsymbol{\tau}_i\}_{i=1}^{N}$. Notably, in sparse-reward open-world applications, process-level rewards r_t are often replaced by trajectory-level rewards $\mathcal{R}^E(\boldsymbol{\tau}_i)$, such as whether the agent achieves the goal at the final step (Shridhar et al., 2020).

Agent Reinforcement Learning (ARL). ARL incorporates full trajectories of agent reasoning and actions (Wang et al., 2025a) into the RL framework. A representative formulation is Group Relative Policy Optimization (GRPO) (Guo et al., 2025); for the task $\boldsymbol{x}$ and sampled rollouts $\{\boldsymbol{\tau}_i\}_{i=1}^{N} \sim \boldsymbol{\pi}_{old}$, GRPO is formulated as the following:

$$\begin{aligned} \mathcal{J}(\theta) &= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\boldsymbol{\tau}_i|}\sum_{t=1}^{|\boldsymbol{\tau}_i|}\min(\rho_{\theta,t}^{(i)}A^E(\boldsymbol{\tau}_i), \\ &\mathrm{clip}(\rho_{\theta,t}^{(i)}, 1\pm\epsilon)A^E(\boldsymbol{\tau}_i)) - \beta\mathcal{D}_{KL}[\boldsymbol{\pi}_\theta||\boldsymbol{\pi}_{ref}] \\ A^E(\boldsymbol{\tau}_i) &= \frac{\mathcal{R}^E(\boldsymbol{\tau}_i) - \mathrm{avg}(\{\mathcal{R}^E(\boldsymbol{\tau}_i)\}_{i=1}^{N})}{\mathrm{std}(\{\mathcal{R}^E(\boldsymbol{\tau}_i)\}_{i=1}^{N})}, \end{aligned} \tag{1}$$

where $\boldsymbol{y}_t^{(i)}$ represents partial trajectory under rollout i at token t, $\rho_{\theta,t}^{(i)} = \boldsymbol{\pi}_\theta(\boldsymbol{y}_t^{(i)}|\boldsymbol{x}, \boldsymbol{y}_{<t}^{(i)}) / \boldsymbol{\pi}_{old}(\boldsymbol{y}_t^{(i)}|\boldsymbol{x}, \boldsymbol{y}_{<t}^{(i)})$ is the importance sampling ratio of $\boldsymbol{y}_t^{(i)}$, rollout i; $\mathcal{D}_{KL}$, $\boldsymbol{\pi}_{old}$ and $\boldsymbol{\pi}_{ref}$ are the KL divergence, old policy and reference policy respectively. β controls the penalty degree of the KL-loss. The following subsections present our proposed Role-Agent, which integrates the World-In-Agent (WIA) and Agent-In-World (AIW) design to achieve the bootstrapped agent-environment co-evolution.

3.2 World-In-Agent (WIA)

Role-Agent first assigns the LLM the role of an agent and requires it to develop fine-grained, interleaved perception of the world. Inspired by world models (Li et al., 2025; Gu et al., 2024), we internalize environment dynamics into the agent by rewarding future-state prediction.

Predicting the Future State. During rollout, at each interaction step t, after the agent generates an action $\boldsymbol{a}_t$, we prompt it with the augmented prompt $\boldsymbol{x}_{pre}$ to predict the future states induced by this action. This encourages the agent to explicitly model how its actions may change the environment, rather than relying only on observed rewards. For each prediction horizon $h \in \{1, \ldots, H\}$, the agent predicts the state at step $t + h$:

$$\hat{\boldsymbol{s}}_{t,h} \sim \boldsymbol{\pi}(\cdot \mid \boldsymbol{a}_t, \boldsymbol{x}_{pre}), \tag{2}$$

where $\hat{\boldsymbol{s}}_{t,h}$ denotes the prediction made at step t for the future state $\boldsymbol{s}_{t+h}$. We denote the prediction set at step t as $\mathcal{E}_{pre,t} = \{\hat{\boldsymbol{s}}_{t,h} \mid 1 \le h \le H\}$, and collect all prediction sets after rollout:

$$\mathcal{E}_{pre} = \{\mathcal{E}_{pre,t} \mid 1 \le t \le T\}. \tag{3}$$

Inspired by GiGPO (Feng et al., 2025), we measure the discrepancy between predicted and ground-truth states using the Longest Matching Subsequence (LMS) over their textual state contexts, yielding a predictive reward matrix $\tilde{\boldsymbol{r}} \in \mathbb{R}^{T\times H}$:

$$\tilde{r}_{t,h} = \mathrm{LMS}(\hat{\boldsymbol{s}}_{t,h}, \boldsymbol{s}_{t+h}), \tag{4}$$

each $\tilde{r}_{t,h} \in [0, 1]$ quantifies the agent's foresight in predicting the state h steps ahead. In implementation, predictive rewards are computed at the end of each rollout. In parallel, we obtain the full trajectory $\boldsymbol{\tau} = \{(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t)\}_{t=1}^{T}$. The task reward for each action $\boldsymbol{a}_t$ is computed as the discounted return from step t, while the predictive reward aggregates future-state prediction scores within horizon H:

$$\mathcal{R}_{task}(\boldsymbol{a}_t) = \sum_{k=t}^{T}\gamma^{k-t}r_k, \mathcal{R}_{pre}(\boldsymbol{a}_t) = \sum_{h=1}^{H}\gamma^{h-1}\tilde{r}_{t,h}. \tag{5}$$

We combine the task and predictive rewards according to two principles: **(a)** accurate state prediction preserves and amplifies the original credit, reflecting reliable environment perception; and **(b)** inaccurate prediction weakens the advantage signal, reducing credit for actions that achieve high returns by chance. Thus, predictive reward serves as a reliability-aware modulation of task reward:

$$\mathcal{R}_t = \mathcal{R}_{task}(\boldsymbol{a}_t)(1 + \mathcal{R}_{pre}(\boldsymbol{a}_t)). \tag{6}$$

We use multiplication rather than addition so that predictive reward cannot independently introduce extra credit. Instead, it only modulates actions with non-zero task reward, preventing failed trajectories from being rewarded solely for plausible state predictions.

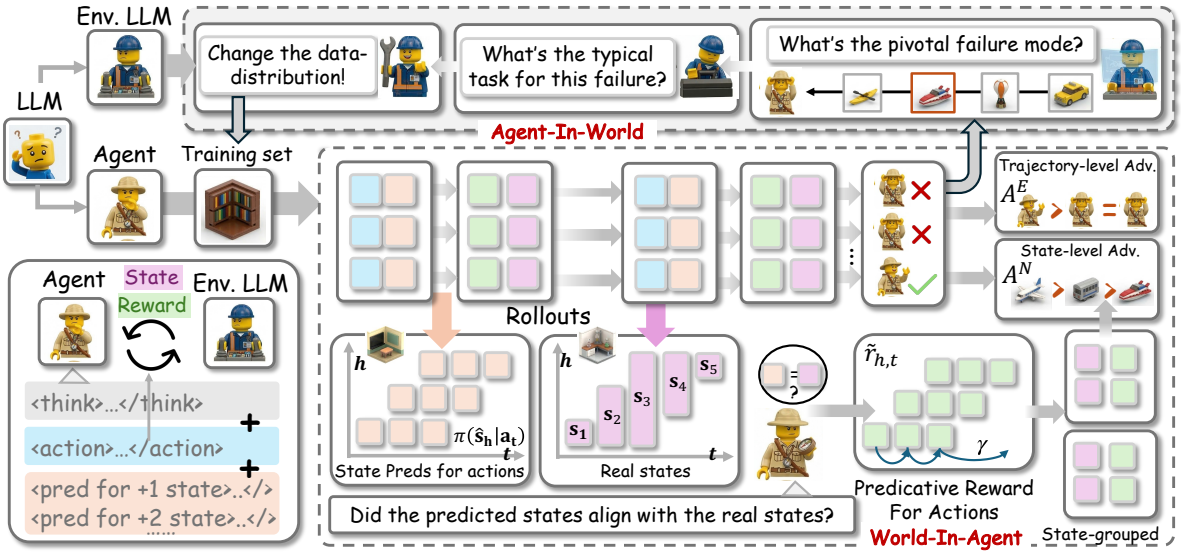

Figure 2: Overview of the Role-Agent. A single LLM is leveraged to switch between the roles of agent and environment. As an agent, it is prompted to predict states for the next H steps; the alignment between these predictions and ground-truth states serves as a reward signal to compute trajectory-level and state-level advantages. As the environment, it analyzes failure modes from failed trajectories and reshapes the data distribution by retrieving tasks with similar modes. This closed-loop process enables bootstrapped agent-environment co-evolution.

State Grouping & State-level Advantage. Instead of employing the trajectory-level advantage, following GiGPO (Feng et al., 2025), we observe that even within the same environment and initial settings, there can be significant redundancy among states in a trajectory. By grouping actions that occur under identical states and computing state-level advantages, we can more clearly attribute rewards at the state level, independent of their temporal ordering. Formally, we identify a set of **non-repetitive states** $\mathcal{O}=\{\boldsymbol{s}_o^{\dagger}\}_{o=1}^{|\mathcal{O}|}$ from the batch with hash-maps, then group the actions like:

$$\mathcal{G}=\{\{(\boldsymbol{s}_t,\boldsymbol{a}_t)|\text{hash}(\boldsymbol{s}_t^{(i)})=\text{hash}(\boldsymbol{s}_o^{\dagger})\}|\boldsymbol{s}_o^{\dagger}\in\mathcal{O}\} \quad (7)$$

Accordingly, we denote $\mathcal{G}^{(o)}=\{(\boldsymbol{s}_t^{(o)},\boldsymbol{a}_t^{(o)})\}$ as the set of state-action pairs grouped by $\boldsymbol{s}_o^{\dagger}$. Finally, the state-level advantage for each $\boldsymbol{a}_t^{(o)}$ is calculated as:

$$A^S(\boldsymbol{a}_t^{(o)})=\frac{\mathcal{R}_t^{(o)}-\text{avg}(\{\mathcal{R}_t^{(o)}|(\boldsymbol{s}_t^{(o)},\boldsymbol{a}_t^{(o)})\in\mathcal{G}^{(o)}\})}{\text{std}(\{\mathcal{R}_t^{(o)}|(\boldsymbol{s}_t^{(o)},\boldsymbol{a}_t^{(o)})\in\mathcal{G}^{(o)}\})} \quad (8)$$

With the state-level advantages, we finally revise the trajectory-level policy optimization of GRPO into the following variants:

$$\mathcal{J}_{ours}(\theta)=\frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\boldsymbol{\tau}_i|}\sum_{t=1}^{|\boldsymbol{\tau}_i|}\min(\rho_{\theta,t}^{(i)}A(\boldsymbol{a}_t^{(i)}),$$
$$\text{clip}(\rho_{\theta,t}^{(i)},1\pm\epsilon)A(\boldsymbol{a}_t^{(i)}))-\beta\mathcal{D}_{KL}[\pi_\theta||\pi_{ref}] \quad (9)$$

where $\rho_{\theta,t}^{(i)}=\boldsymbol{\pi}_\theta(\boldsymbol{a}_t^{(i)}|\boldsymbol{s}_t^{(i)},\boldsymbol{y}_{<t}^{(i)})/\boldsymbol{\pi}_{old}(\boldsymbol{a}_t^{(i)}|\boldsymbol{s}_t^{(i)},\boldsymbol{y}_{<t}^{(i)})$ is the importance sampling ratio at step t for rollout i. The advantage is derived from the trajectory-level and state-level advantages, linked with coefficient α, i.e., $A(\boldsymbol{a}_t^{(i)})=A^S(\boldsymbol{a}_t^{(o)})+\alpha\cdot A^E(\boldsymbol{\tau}_i)$, where o denotes the group to which $\boldsymbol{a}_t^{(i)}$ belongs.

3.3 Agent-In-World (AIW)

Beyond enabling the agent to perceive world dynamics, we argue that the environment should also dynamically adjust itself based on the agent's capability. To this end, we propose Agent-In-World (AIW), which allows the agent itself to act as a source of environmental feedback. By receiving, validating, and filtering its own interaction history, the agent expands the data distribution in a self-regulated manner.

Failure Mode Analysis. For each failed trajectory, we feed all interaction sequences, along with the task description and objective, into an LLM for analysis. We prompt the LLM to identify one or more action patterns that led to the failure, and to generate a failure-mode reflection that includes the failure type, core lessons, and query contexts to be used for retrieving similar tasks subsequently.

Task Retrieval & Changing Data Distribution. Subsequently, we store these failure modes along with the corresponding failed trajectories and task information in an offline interaction history. The entire history of failure modes is then fed into the LLM, which is instructed to retrieve patterns similar to the current failure mode and return the indices

of relevant interaction histories. In practice, we organize tasks under unique failure modes rather than referring to every failed trajectory. On ALFWorld, this library comprises 11 unique modes across training, and storage or retrieval costs remain negligible. Tasks grouped by shared failure modes highlight the LLM's deficiencies and oversights when facing specific situations. Accordingly, we reintegrate these retrieved tasks into the training set. Compared with random failed-task replay or task-text retrieval, AIW retrieves by the underlying error pattern, which can connect surface-different tasks sharing the same procedural weakness. By using the same LLM to switch roles, we establish an agent-environment co-evolution without introducing a separate model in the fine-tuning stage.

4 Experiments

4.1 Experiment Setups

Benchmarks. We evaluate our method across three types of tasks: ALFWorld (Shridhar et al., 2020), WebShop (Yao et al., 2022a), and search-augmented question answering (QA). ALFWorld assesses the model's multi-step decision-making abilities through household tasks, where they are required to navigate the environment using textual commands to achieve given goals. WebShop is a simulated e-commerce platform where agents interact with a realistic web interface containing over 1.18 million real-world products. Additionally, we employ search-augmented QA tasks, which include single-hop QA datasets such as NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2023), as well as multi-hop QA datasets including HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023). Together, these benchmarks enable a comprehensive evaluation of an agent's ability to ground language while effectively leveraging external information.

Baselines. We compare Role-Agent with various competitive models, categorized as follows: **(a)** Closed-source models: GPT-4o (Achiam et al., 2023) and Gemini-2.5-Pro (Team et al., 2023), which achieve superior performance in general-purpose reasoning and understanding. **(b)** Prompt engineering methods: ReAct (Yao et al., 2022b) and Reflexion (Shinn et al., 2023), which leverage prompts to structure the multi-step behavior of agents. **(c)** RL training methods: PPO (Schulman et al., 2017), which utilizes the collaboration between actor and critic networks along with a reward model; RLOO (Kool et al., 2019; Ahmadian et al., 2024) and GRPO (Guo et al., 2025), which compute advantages within grouped trajectories. **(d)** Search-based models (evaluated only on search-QA tasks): R1-Instruct (Jin et al., 2025), Search-R1 (Jin et al., 2025), ZeroSearch (Sun et al., 2025), and StepSearch (Wang et al., 2025b).

Implementation Details. We employ Qwen2.5-1.5/3B/7B-Instruct as backbone models for all experiments. All baselines adopt the same hyperparameters (if shared) values as our method. Following (Feng et al., 2025), the group size for RLOO and GRPO is set to 8. For the search tasks we use E5 as the retriever, with a group size of 5 and a maximum of 4 turns. All models are trained on a single node with 8 NVIDIA H20 GPUs. State grouping is performed by merging states whose longest-matching subsequence similarity exceeds 0.9. We keep this threshold from GiGPO for fair comparison; Also, since states of all datasets we employed are short and templated, a high threshold avoids the conflation of genuinely different states. The maximal steps T_{max} for ALFWorld, WebShop and Search QA are 50, 15, 4 respectively. Details of datasets, implementations and prompts are in **Appendix A/C/D** respectively.

4.2 Experimental Results

Results on ALFWorld and WebShop. Table 1 presents the comparison results, showing that Role-Agent consistently outperforms various existing baselines. Our key observations are as follows:

(a): Traditional prompt-based methods such as ReAct and Reflexion yield considerable gains over zero-shot models but still underperform compared to RL-based methods. Role-Agent, in particular, outperforms these approaches by an average of 78.0% on ALFWorld and 59.1% on WebShop in terms of success rate. While closed-source models like Gemini achieve competitive performance on specific tasks (e.g., 92.8% on Pick-ALFWorld), their average performance lags behind, underscoring both the difficulty of the tasks and the benefits of post-training. This suggests that prompts can enhance the in-context learning ability of agents but do not enable internal adaptation.

(b): RL training methods yield substantial gains, as demonstrated by GRPO which achieves 72.8%

Type	Method	ALFWorld							WebShop	
		Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Succ.
Closed-source Model										
Prompting	GPT-4o	75.3	60.8	31.2	56.7	21.6	49.8	48.0	31.8	23.7
Prompting	Gemini-2.5-Pro	92.8	63.3	62.1	69.0	26.6	58.7	60.3	42.5	35.9
Qwen2.5-1.5B-Instruct										
Prompting	Qwen-2.5	5.9	5.5	3.3	9.7	4.2	0.0	4.1	23.1	5.2
Prompting	ReAct	17.4	20.5	15.7	6.2	7.7	2.0	12.8	40.1	11.3
Prompting	Reflexion	35.3	22.2	21.7	13.6	19.4	3.7	21.8	55.8	21.9
RL Training	PPO	64.8	40.5	57.1	60.6	46.4	47.4	54.4	73.8	51.5
RL Training	RLOO	88.3	52.8	71.0	62.8	66.4	56.9	69.7	73.9	52.1
RL Training	GRPO	85.3	53.7	84.5	78.2	59.7	53.5	72.8	75.8	56.8
RL Training	GiGPO	94.4	67.5	94.8	94.4	79.8	76.4	86.7	83.1	65.0
RL Training	**Role-Agent**	**95.8**	**78.3**	**95.0**	**97.0**	**87.5**	**91.7**	**90.9**	**87.7**	**71.9**
Qwen2.5-7B-Instruct										
Prompting	Qwen-2.5	33.4	21.6	19.3	6.9	2.8	3.2	14.8	26.4	7.8
Prompting	ReAct	48.5	35.4	34.3	13.2	18.2	17.6	31.2	46.2	19.5
Prompting	Reflexion	62.0	41.6	44.9	30.9	36.3	23.8	42.7	58.1	28.8
RL Training	PPO	92.3	64.0	92.5	**89.5**	80.3	68.8	80.4	81.4	68.7
RL Training	RLOO	87.6	78.2	87.3	81.3	71.9	48.9	75.5	80.3	65.7
RL Training	GRPO	90.8	66.1	89.3	74.7	72.5	64.7	77.6	79.3	66.1
RL Training	GiGPO	97.7	82.7	**98.8**	83.7	89.3	79.2	90.8	84.4	72.8
RL Training	**Role-Agent**	**98.3**	**93.7**	98.5	88.9	**90.0**	**92.8**	**93.8**	**88.0**	**77.1**

Table 1: Performance comparison on ALFWorld and WebShop. We report the average success rate (%) for each task and the averaged performance in ALFWorld; For WebShop, we report the averaged score and success rate (%).

Type	Method	Single-Hop QA			Multi-Hop QA				Avg.
		NQ†	TriviaQA*	PopQA*	HotpotQA†	2Wiki*	MuSiQue*	Bamboogle*	
RL Training	R1-Instruct	27.0	53.7	19.9	23.7	29.2	7.2	29.3	27.1
RL Training	Search_R1	34.1	54.5	37.8	32.4	31.9	10.3	26.4	32.5
RL Training	Zero-Search	41.4	57.4	44.8	27.4	30.0	9.8	11.1	31.7
RL Training	StepSearch	-	-	-	34.5	32.0	17.4	34.4	-
RL Training	GiGPO	**42.0**	59.5	42.4	36.9	37.0	12.6	64.1	42.1
RL Training	**Role-Agent**	40.1	**60.4**	**49.8**	**38.8**	**45.2**	**17.8**	**68.4**	**45.8**

Table 2: Comparison on search-augmented QA tasks. Role-Agent is trained on NQ and HotpotQA. † and $*$ indicate in-domain and out-of-domain datasets, respectively. All methods are experimented with Qwen2.5-3B-Instruct.

/ 75.8% on ALFWorld / WebShop, and GiGPO which achieves 86.7% / 83.1% respectively with Qwen2.5-1.5B-Instruct. The success of GiGPO stems from its group-level advantages, which unify action evaluation across different steps for the same state. Nevertheless, Role-Agent mostly outperforms GiGPO across both backbone models, with relative gains of 4.2% / 6.9% on two datasets, validating that the co-evolution in Role-Agent equips the agent with more generalization abilities. Stability results are in Appendix B.

(c): Role-Agent demonstrates consistent superiority across larger backbone models (Qwen2.5-7B-Instruct), achieving average gains of 3.8%. Moreover, improvements are more pronounced in complex and compositional tasks. For instance, Role-Agent shows a +11.0% increase on the Look task (i.e., look_at_obj_in_light) and a +13.6% increase on the Pick2 task (i.e., pick_two_obj_and_place), both of which require stable memory and multi-step planning to ensure the correctness of each subtask. These results further indicate that the bootstrapped agent-environment co-evolution endows agents with substantial generalization capabilities.

Method	ALFWorld	WebShop	Average
Role-Agent	**90.9**	**71.9**	**81.4**
- *w/o Agent-In-World*	87.5	66.9	77.2
- *w/o Predictive Reward*	88.0	68.3	78.2
GiGPO	86.7	65.0	75.9

Table 3: Ablation study of components with Qwen2.5-1.5B-Instruct. We report the average success rate.

Results on Search-Augmented QA tasks. Table 2 presents the results. Role-Agent achieves the best average performance of 45.8%, outperforming the GiGPO average by 3.7%. Notably, the performance gains are more pronounced on multi-hop QA tasks compared to single-hop ones, with improvements of +8.2% on 2Wiki and +5.2% on MuSiQue. These results demonstrate that agent-environment co-evolution equips the agent with enhanced multi-step retrieval and reasoning capabilities. We also observe that Role-Agent slightly underperforms GiGPO on NQ dataset, which we attribute to its stronger generalization capabilities rather than overfitting to the training set. Since search-agent baselines differ in training and eval-

Hyper-Param.	Value	ALFWorld	WebShop	Average
Adv. Scaling Coef. α	0.5	89.5	71.0	80.2
	1.0	**90.9**	**71.9**	**81.4**
	2.0	86.0	65.4	75.7
# Prediction Step H	$5\% \cdot T_{max}$	**90.9**	**71.9**	**81.4**
	$10\% \cdot T_{max}$	90.2	68.5	79.3
	$20\% \cdot T_{max}$	75.6	62.3	69.0

Table 4: Sensitivity Analysis of hyper-parameters with Qwen2.5-1.5B-Instruct. ($H \geq 1$ and is rounded).

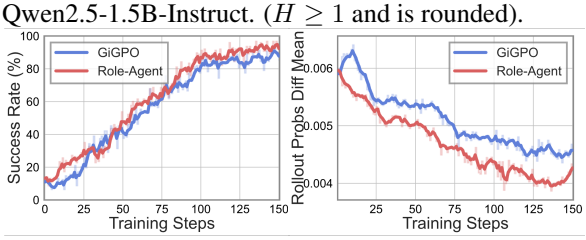

Figure 3: Running dynamics on ALFWorld. **(left):** success rate on the validation set; **(right):** the averaged difference between training and inference rollouts.

uation protocols, we use these results as a cross-domain comparison and rely on ALFWorld/WebShop comparisons for the most direct assessment.

4.3 Ablation Study & Sensitivity Analysis

Effects of Components. We conduct an ablation study by comparing the performance of Role-Agent against variants with specific components removed. The results are presented in Table 3. Specifically, we find that removing either the AIW module or the predictive reward leads to a drop in overall performance, with the effect being more pronounced for AIW removal (a 5.0% decrease on WebShop). This highlights the pivotal role of targeted environment feedback in AIW. Without dynamic data distribution, the agent lacks iterative practice on critical failure modes. The results also confirm that the predictive reasoning in WIA equips the agent with valuable implicit world priors, enhancing its decision-making capabilities at every step. Notably, both ablated variants still outperform GiGPO on average, indicating that WIA and AIW each provide gains beyond state-grouped credit assignment alone and are complementary rather than redundant.

Failure Mode Evolution. Figure 4 visualizes the cumulative evolution of failure modes during training. The total number of recorded failures grows quickly in the early stage and then gradually saturates, from 996 at step 15 to 3931 at step 150. This suggests that tasks are assigned to failure-mode buckets rapidly at first, and per-mode accumulation then tapers off as the library becomes sufficiently populated. Among different categories, repetitive exploration, wrong target location, and wrong receptacle account for a large proportion of failures, suggesting the importance of exploration

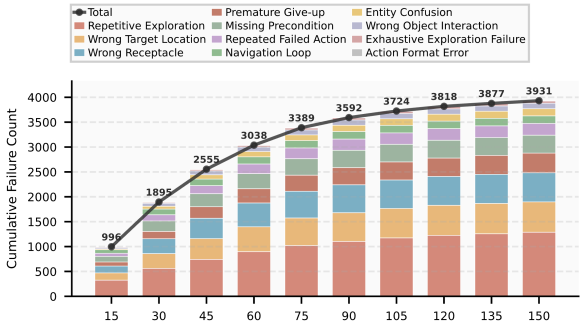

Figure 4: Tasks of failure modes accumulated in training. Tracked on ALFWorld with Qwen2.5-7B-Instruct.

and grounding. The increment of other modes show that AIW captures diverse and fine-grained weaknesses rather than a single dominant error type. These results show that by accumulating structured failure modes over training, the environment can provide more targeted tasks for the agent to revisit its historical deficiencies. Failure modes of all datasets are provided in Appendix C.

Hyper-parameter Sensitivity. We vary the advantage scaling coefficient α and the number of steps per prediction H, and report the results in Table 4. When studying one hyperparameter, the other is fixed at its optimal value. Our findings are:

(a): The coefficient α controls the balance between trajectory-level and state-level advantages in the final optimization signal. When $\alpha = 0.5$, Role-Agent achieves 89.5% on ALFWorld and 71.0% on WebShop, which is slightly lower than the default setting. This indicates that under-weighting the trajectory-level advantage may weaken the global task-completion signal. When α is increased to 2.0, the average performance drops clearly to 75.7%, suggesting that excessive trajectory-level weighting can dilute the fine-grained state-level credit assignment. Therefore, setting $\alpha = 1.0$ provides a balanced integration of both advantage terms and achieves the best average performance.

(b) Increasing H beyond $5\% \cdot T_{max}$ (T_{max} is the number of maximal interaction steps) generally leads to a sharp decline in performance. For instance, at H=$10\% \cdot T_{max}$, the average accuracy on WebShop drops to 68.5%; further increases eventually render Role-Agent ineffective, causing it to underperform most RL methods. This degradation may be attributed to that excessive predictions occupy the in-context window and diminish the agent's focus on action planning. Additionally, predicting states too far beyond the current context can lead to speculative guesswork and reward hacking. Therefore, setting H=$5\% \cdot T_{max}$ achieves a Pareto

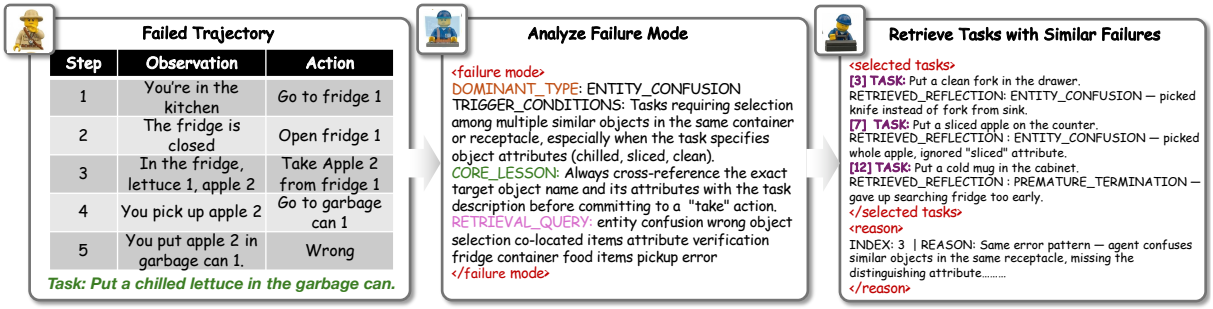

Figure 5: Case study of Agent-In-World in Role-Agent on ALFWorld, illustrating how the environment LLM extracts failure modes from failed trajectories and retrieves tasks with similar failure modes.

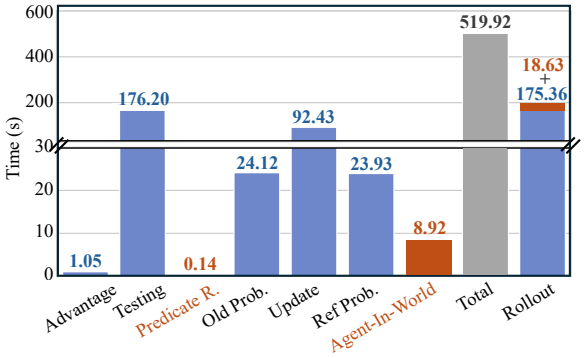

Figure 6: Per-step time breakdown of Role-Agent. *The gray bar represents the average time of a complete generation.* The blue bar indicates the runtime of the comparative baseline (GiGPO), while **the orange bars highlight the additional runtime from our method.**

optimum in both efficiency and effectiveness.

Running Dynamics. Figure 3 compares the running curves of Role-Agent and GiGPO. In Figure 3 **(left)**, we find that while Role-Agent falls behind GiGPO or shows fluctuation in the beginning stage, it generally achieves a higher performance ceiling (90.9%) and faster convergence. This suggests that the effects of closed-loop agent-environment evolution intensify with the accumulation of adjusted data distribution, where the agent receives targeted training on its failures. In Figure 3 **(right)**, we plot the difference between training and inference rollouts. Compared with GiGPO, Role-Agent brings a substantial mitigation of the train-inference mismatch. Higher consistency between the rollout and training policies leads to lower variance in gradient estimation and improves training stability.

Efficiency Study. Figure 6 compares the running time of different components, with Role-Agent-specific costs highlighted in orange. All of the efficiency results are evaluated on ALFWorld.

The extra predictions during rollout, calculations of predictive reward and Agent-In-World feedback are minor (18.63s, 0.14s, 8.92s) compared with the overall running-time per step, inducing only about 5.2% extra computation. The state comparison contains only the task description and two short state descriptions, and the retrieval repository contains only a small number of unique failure modes. Additional retrieved tasks alter the sampling distribution but do not require a separate model. Together with the gains in Table 1, these results show that Role-Agent balances efficiency and effectiveness.

4.4 Case Study

Figure 5 further illustrates how the environment LLM adjusts the data distribution by analyzing failure trajectories. In the shown failed trajectory, the agent mistakenly picks "Apple 2" from the fridge in step 3. The environment LLM then identifies the failure mode as ENTITY_CONFUSION, along with a description of how the failure occurred and queries for retrieving similar failure modes. Finally, it searches for analogous failures in the stored history of (task, failure mode) pairs. This workflow shows how structured failure analysis enables more targeted subsequent training.

5 Conclusion

This paper introduces Role-Agent, a bootstrapped framework for agent-environment co-evolution designed to overcome the challenges of undirected and non-specific feedback in static environments. Role-Agent leverages a single Large Language Model (LLM) to act as both the agent and the environment, realized through our World-In-Agent (WIA) and Agent-In-World (AIW). WIA enhances the agent's planning and reasoning by equipping it with the ability to predict future states based on its actions; AIW uses the same LLM to analyze failure patterns from unsuccessful trajectories and retrieve analogous tasks, thereby dynamically reshaping the training data distribution. Extensive experiments across diverse benchmarks validate that Role-Agent achieves strong performance, demonstrating the effectiveness of our approach.

Limitations

Despite its effectiveness, Role-Agent has several limitations. First, a stronger frozen environment LLM can improve the AIW component, but it also introduces extra external knowledge and changes the fairness of comparison with same-backbone baselines. Second, the state grouping mechanism employs a similarity threshold from previous studies, limiting cross-task generalization. Finally, the current evaluation is confined to text-based environments; extensions to multi-modal or real-time embodied settings may require vision-language state descriptions or latent-state matching and remain important future work.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. 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)*, pages 12248–12267.

AI Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 1(1):4.

Ke Chen, Peiran Wang, Yaoning Yu, Xianyang Zhan, and Haohan Wang. 2025a. Large language model-based data science agent: A survey. *arXiv preprint arXiv:2508.02744*.

Yixing Chen, Yiding Wang, Siqi Zhu, Haofei Yu, Tao Feng, Muhan Zhang, Mostofa Patwary, and Jiaxuan You. 2025b. Multi-agent evolve: Llm self-improve through co-evolution. *arXiv preprint arXiv:2510.23595*.

Xiangxiang Chu, Hailang Huang, Xiao Zhang, Fei Wei, and Yong Wang. 2026. GPG: A simple and strong reinforcement learning baseline for model reasoning. In *The Fourteenth International Conference on Learning Representations*.

Dave Citron. 2024. Try deep research and our new experimental model in gemini, your ai assistant. *Google Blog, December*, 11.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. 2025a. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning. *arXiv preprint arXiv:2505.16410*.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, and 1 others. 2025b. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. 2025. Memp: Exploring agent procedural memory. *arXiv preprint arXiv:2508.06433*.

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

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2023. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, and 1 others. 2025. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*.

Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, and 1 others. 2024. Is your llm secretly a world model of the internet? model-based planning for web agents. *arXiv preprint arXiv:2411.06559*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Erik Hemberg, Stephen Moskal, and Una-May O'Reilly. 2024. Evolving code with a large language model. *Genetic Programming and Evolvable Machines*, 25(2):21.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.

Shengran Hu, Cong Lu, and Jeff Clune. 2024. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435*.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. 2025. Aide: Ai-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*.

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

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Wouter Kool, Herke van Hoof, and Max Welling. 2019. Buy 4 reinforce samples, get a baseline for free! *ICLR 2019 Workshop*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, and 1 others. 2024. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Junlong Li, Daya Guo, Dejian Yang, Runxin Xu, Yu Wu, and Junxian He. 2025. Codei/o: Condensing reasoning patterns via code input-output prediction. *arXiv preprint arXiv:2502.07316*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, and 1 others. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Ziyu Ma, Shidong Yang, Yuxiang Ji, Xucong Wang, Yong Wang, Yiming Hu, Tongwen Huang, and Xiangxiang Chu. 2026. Skillclaw: Let skills evolve collectively with agentic evolver. *arXiv preprint arXiv:2604.08377*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 9802–9822.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, and 1 others. 2025. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*.

Yixin Ou, Yujie Luo, Jingsheng Zheng, Lanning Wei, Zhuoyun Yu, Shuofei Qiao, Jintian Zhang, Da Zheng, Yuren Mao, Yunjun Gao, and 1 others. 2025. Automind: Adaptive knowledgeable agent for automated data science. *arXiv preprint arXiv:2506.10974*.

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

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

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

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Tongyi DeepResearch Team, Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, Jialong Wu, Jingren Zhou, and 1 others. 2025. Tongyi deepresearch technical report. *arXiv preprint arXiv:2510.24701*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. *Advances in Neural Information Processing Systems*, 37:2686–2710.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, and 1 others. 2025a. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*.

Ziliang Wang, Xuhui Zheng, Kang An, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu. 2025b. Stepsearch: Igniting llms search ability via stepwise proximal policy optimization. *arXiv preprint arXiv:2505.15107*.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, and 1 others. 2025. Evolver: Self-evolving llm agents through an experience-driven lifecycle. *arXiv preprint arXiv:2510.16079*.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, and 1 others. 2026. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*.

Taofeng Xue, Chong Peng, Mianqiu Huang, Linsen Guo, Tiancheng Han, Haozhe Wang, Jianing Wang, Xiaocheng Zhang, Xin Yang, Dengchang Zhao, and 1 others. 2026. Evocua: Evolving computer use agents via learning from scalable synthetic experience. *arXiv preprint arXiv:2601.15876*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022a. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, and 1 others. 2025. Agentevolver: Towards efficient self-evolving agent system. *arXiv preprint arXiv:2511.10395*.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. 2025a. Memevolve: Meta-evolution of agent memory systems. *arXiv preprint arXiv:2512.18746*.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. 2026a. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, and 1 others. 2026b. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*.

Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, and 1 others. 2025b. Process vs. outcome reward: Which is better for agentic rag reinforcement learning. *arXiv preprint arXiv:2505.14069*.

Zhuosheng Zhang and Aston Zhang. 2024. You only look at screens: Multimodal chain-of-action agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3132–3149.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, and 1 others. 2025.

Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

Terry Yue Zhuo, Dingmin Wang, Hantian Ding, Varun Kumar, and Zijian Wang. 2025. Cyber-zero: Training cybersecurity agents without runtime. *arXiv preprint arXiv:2508.00910*.

A Dataset Details

A.1 ALFWorld

ALFWorld is an interactive framework that bridges text-based environments and physically embodied simulations. Agents learn high-level policies in TextWorld and apply them within the visual ALFRED benchmark. With parallel representations of the same world, ALFWorld enables agents to leverage semantic priors and language-based reasoning to generalize more effectively to new tasks. This dual-modality design promotes stronger generalization and greater training efficiency compared to vision-only approaches.

A.2 WebShop

WebShop is a large-scale simulated e-commerce environment comprising over 1.18 million real-world products and 12,087 crowd-sourced natural language instructions for training grounded language agents. In this benchmark, agents use two actions, i.e., search[query] and click[element], to fulfill complex user requirements. The environment features an automatically computable reward function based on product attributes, which shows sim-to-real transfer capabilities when deployed on actual shopping websites like Amazon and eBay.

A.3 Search-QA Tasks

Natural Questions (NQ) is a large-scale open-domain QA dataset built from real Google search queries. Each example pairs a user question with an answer extracted from a Wikipedia page. It's widely used to benchmark single-hop retrieval and reading comprehension, where the task is to locate and extract an answer from a single passage.

TriviaQA consists of question-answer-evidence triples sourced from Wikipedia and news articles. The questions involve complex entity relationships, and the evidence is collected via distant supervision, meaning the answer isn't necessarily tied to a single pre-selected passage. It's commonly used to test how well models retrieve and synthesize facts from unstructured text.

PopQA focuses on long-tail entities. It samples over 14,000 questions about less frequently mentioned entities. It tests whether a retrieval method can actually look up obscure knowledge instead of relying on parametric memory.

HotpotQA requires multi-hop reasoning across two or more Wikipedia paragraphs. It's designed to test whether a model can follow a chain of evidence, not just locate a single fact.

2WikiMultihopQA (2Wiki) is constructed using a rule-based template system. This ensures each question has a predefined reasoning path, and questions are categorized by logical type, such as comparison, temporal, or compositional. It provides a controlled setting for evaluating whether models can perform specific kinds of multi-step inference.

MuSiQue is built by programmatically composing single-hop questions from existing datasets like SQuAD and TriviaQA. The composition ensures strict connectivity between reasoning steps. It includes unanswerable distractors. This tests whether models can follow a dependency chain while filtering out irrelevant information.

Bamboogle is designed to be unsolvable by parametric models alone. The questions require decomposition and sequential retrieval across multiple documents. It's used to evaluate whether search-augmented agents can generalize compositionally, meaning they can combine facts from different sources in ways that weren't seen during training.

B More Studies

B.1 Standard Deviations

Table 7 reports the mean and standard deviations over three runs.

B.2 Relation between predictive reward and outcome reward

On 200 ALFWorld rollouts with Qwen2.5-3B-Instruct, the predictive reward has a point-biserial correlation of 0.41 ($p < 0.01$) with outcome reward. Its average value also rises from about 0.60 at initialization to the mid-to-high 0.70 range near convergence, indicating improved state prediction quality.

C Implementation Details

Role-Agent adopts the VeRL framework to train agents. We list the detailed hyper-parameters in Table 5. All of the employed backbones in the experiments, i.e., Qwen2.5-1.5/3/7B-Instruct are

Parameter	Value	Parameter	Value
learning rate	1.00E-06	evaluation temperature	0
training batch size	16/16/256	max response length	4096
optimizer	AdaW	reward	suc=1,fail=0
clip ratio low	0.2	max interaction step	50/15/4
clip ratio high	0.28	state similarity threshold	0.9
KL coefficient	1.00E-03	$reflectiontemperature$	0.5
rollout temperature	0.9	total epoch	150
val_data_size	128/128/512	group_size	8

Table 5: Hyper-parameters for RL training.

ALFWorld	**WebShop**	**Search**
repetitive_exploration	irrelevant_query	wrong_answer
wrong_target_location	wrong_product_selection	insufficient_retrieval
wrong_receptacle	wrong_attribute_selection	irrelevant_retrieval_query
premature_give_up	missing_attribute_selection	repeated_retrieval_query
missing_precondition	premature_purchase	information_misinterpretation
repeated_failed_action	excessive_browsing	partial_answer
navigation_loop	repeated_query	hallucinated_answer
entity_confusion	navigation_error	premature_answer
wrong_object_interaction	price_constraint_violation	action_format_error
exhaustive_exploration_failure	action_format_error	
action_format_error	premature_termination	

Table 6: Failure modes used in Agent-In-World across ALFWorld, WebShop, and search-augmented QA tasks.

Method	**ALFWorld**	**WebShop**
Qwen2.5-1.5B-Instruct		
GRPO	72.8 ± 1.5	56.8 ± 0.7
GiGPO	86.7 ± 0.6	65.0 ± 1.1
Role-Agent	90.9 ± 0.8	71.9 ± 0.9
Qwen2.5-7B-Instruct		
GRPO	77.6 ± 1.0	66.1 ± 0.9
GiGPO	90.8 ± 0.5	72.8 ± 1.8
Role-Agent	93.8 ± 0.8	77.1 ± 0.6

Table 7: Stability results over three runs with Qwen2.5-1.5B/7B-Instruct.

trained on 8× NVIDIA H20 GPUs with tensor parallel equals 1. The list of failure modes employed by Role-Agent is shown in Table 6.

D Prompts

We provide all the prompts we used in the experiment in Figure 7 to 9.

To be specific, Figure 7 shows the specific prompt for search-augmented QA tasks, where we provide the history of interaction, search query and corresponding results. We ask the LLM to either search the website or answer the question. The prompt in Figure 8 first feeds the LLM with the task context and failed trajectories, then asks the LLM to generate typical failure modes, including failure categories, core lessons and suggested queries for the incoming retrieval. The prompt in Figure 9 takes the generated content and asks the LLM to retrieve tasks with similar failure modes, which are stored in the offline library.

E Algorithm

The algorithm is listed in Algorithm 1.

F The Use of Large Language Models

During manuscript preparation, we use large language models (LLMs) to (i) improve grammar and spelling without altering the intended scientific content, and (ii) provide lightweight coding assistance (e.g., scripts and formatting help). All reported numerical results, analyses, and claims are produced by the authors. The authors design the methods, conduct the experiments, and verify the findings.

Prompt Template for Search

You are an expert assistant whose task is to answer the given question step by step.
Your question: {task_description}. So far, you have completed step_count step(s). Below is the interaction history, where <search> and </search> enclose your previous search queries, and <information> and </information> enclose the corresponding results returned by the external search engine. History: memory_context
Now it is your turn to respond at the current step. Begin by conducting your reasoning process. This reasoning must be enclosed within <think> and </think> tags.
After reasoning, choose only one of the following actions (do not attempt both):
(1) If you determine that you are missing some necessary information, you may use a search engine to obtain more external knowledge by formatting your query as: <search> your query </search>.
(2) If you have sufficient knowledge to confidently answer the question, provide your final answer enclosed within <answer> and </answer> tags, without any detailed explanation.

Figure 7: The prompt template of Search agents.

Prompt Template for Abstracting Failure Modes from Failed Trajectories

You are an expert AI trainer specializing in diagnosing why AI agents fail at multi-step reasoning tasks.
Task Context
{task_description}
Failed Trajectory
The agent attempted the task above but failed. Here are the steps it took:
{trajectory_description}
Your Analysis Task Carefully examine the trajectory and produce a structured failure analysis.
Step 1 – Root Cause Identification
Identify the PRIMARY failure modes and describe briefly:
Step 2 – Critical Step Identification
Identify the SINGLE step where the failure became irreversible (the "point of no return").
Step 3 – Core Lesson
State a concise, generalizable lesson (1-2 sentences) that would help an agent avoid this class of mistake on SIMILAR tasks in the future. Focus on the decision rule, not the specific content.
Output Format
Wrap your entire analysis in <reflection> tags using this exact structure:
<reflection>
DOMINANT_TYPE: [category from Step 1]
DETAIL: [1-2 sentences explaining why this root cause applies]
CRITICAL_STEP: [step number and brief description of what went wrong]
CORE_LESSON: [the generalizable rule an agent should follow]
RETRIEVAL_QUERY: [the query for the retrieval stage]
</reflection>

Figure 8: The prompt template for abstracting failure modes from failed trajectories.

Algorithm 1 Role-Agent Training

Require: Initial policy π_θ, reference policy π_{ref}, task pool $\mathcal{D}$, prediction horizon H, discount factor γ, mixing coefficient α

Ensure: Optimized policy π_θ

Initialize task distribution $p_{\mathcal{D}}$ and failure memory $\mathcal{M} \leftarrow \emptyset$
for each training iteration **do**
  Sample a batch of tasks $\{q_i\}_{i=1}^{N} \sim p_{\mathcal{D}}$
  for each task q_i **do**
    Roll out the LLM agent to obtain trajectory $\boldsymbol{\tau}_i = \{(\boldsymbol{s}_t^{(i)}, \boldsymbol{a}_t^{(i)}, r_t^{(i)})\}_{t=1}^{T_i}$
    for each step t in $\boldsymbol{\tau}_i$ **do**
      Use the same LLM with prompt $\boldsymbol{x}_{pre}$ to predict future states $\{\hat{\boldsymbol{s}}_{t,h}^{(i)}\}_{h=1}^{H}$
      Compute predictive scores $\tilde{r}_{t,h}^{(i)} = \text{LMS}(\hat{\boldsymbol{s}}_{t,h}^{(i)}, \boldsymbol{s}_{t+h}^{(i)})$
      Compute task and predictive rewards:

$$\mathcal{R}_{task}^{(i)}(\boldsymbol{a}_t) = \sum_{k=t}^{T_i} \gamma^{k-t} r_k^{(i)}, \quad \mathcal{R}_{pre}^{(i)}(\boldsymbol{a}_t) = \sum_{h=1}^{H} \gamma^{h-1} \tilde{r}_{t,h}^{(i)}$$

      Modulate the reward:

$$\mathcal{R}_t^{(i)} = \mathcal{R}_{task}^{(i)}(\boldsymbol{a}_t)\big(1 + \mathcal{R}_{pre}^{(i)}(\boldsymbol{a}_t)\big)$$

    end for
  end for
  Group identical states across the rollout batch using hash maps
  Compute state-level advantage $A_o^S(\boldsymbol{a}_t^{(i)})$ within each state group $\mathcal{G}_o$
  Compute the final advantage:

$$A(\boldsymbol{a}_t^{(i)}) = A_o^S(\boldsymbol{a}_t^{(i)}) + \alpha A^E(\boldsymbol{\tau}_i)$$

  Update π_θ with the GRPO-style clipped objective using $A(\boldsymbol{a}_t^{(i)})$
  for each failed trajectory $\boldsymbol{\tau}_i$ **do**
    Use the same LLM as the environment role to analyze failure causes
    Generate failure mode and reflection, then store them in $\mathcal{M}$
  end for
  Retrieve tasks in $\mathcal{D}$ similar to the accumulated failure modes
  Update $p_{\mathcal{D}}$ to prioritize difficult and overlooked tasks
end for
return π_θ

Prompt Template for Retrieving Tasks with Similar Failure Modes

You are an expert AI curriculum designer. Your job is to identify which historical training tasks are most relevant for helping an agent overcome a specific failure pattern.
Current Failure Pattern
The agent is currently struggling with the following error pattern:
error_pattern
Historical Task Candidates
Below are historical tasks where the agent previously failed. Each entry shows the task description and a brief failure analysis.
candidates_text
Your Task
Select tasks from the list above that are MOST SIMILAR to the current failure pattern.
Similarity means:
1. The task requires the same type of reasoning or skill that the agent is currently failing at.
2. The task's failure analysis describes a similar root cause or mistake.
3. Re-training on this task would most directly help the agent overcome the current pattern.
Output Format
Output ONLY the following structured block, with no additional text:
<selected_tasks>
INDEX/TASK/REFLECTIONS: <index, task and reflections from the candidate list>
REASON: <one sentence explaining why this task matches the current failure pattern>
INDEX/TASK/REFLECTIONS: <index, task and reflections from the candidate list>
REASON: <one sentence explaining why this task matches the current failure pattern>
</selected_tasks>

Figure 9: The prompt template for retrieving tasks with similar failure modes.

OBSERVATION (step 1)

"You are in the middle of a room. Looking quickly around you, you see a cabinet 1, a cabinet 2, a countertop 1, a garbagecan 1, a handtowelholder 1, a sinkbasin 1, a toilet 1, a toiletpaperhanger 1, and a towelholder 1. Your task is to: put a clean soapbar in toiletpaperhanger."

AGENT_ACTION (step 1)

"go to cabinet 1".

OBSERVATION (step 2)

"You open the cabinet 1. The cabinet 1 is open. In it, you see a cloth 1."

AGENT_ACTION (step 2)

"go to cabinet 2".

OBSERVATION (step 3)

"You open the cabinet 2. The cabinet 2 is open. In it, you see a soapbar 1."

AGENT_ACTION (step 3)

"take soapbar 1 from cabinet 2".

OBSERVATION (step 4)

"You pick up the soapbar 1 from the cabinet 2."

AGENT_ACTION (step 4)

"go to toiletpaperhanger 1"

OBSERVATION (step 5)

"You arrive at toiletpaperhanger 1."

AGENT_ACTION (step 5)

"put soapbar 1 in/on toiletpaperhanger 1"

OBSERVATION (step 6)

"You put the soapbar 1 in/on the sinkbasin 1."

AGENT_ACTION (step 6)

"clean soapbar 1 with sinkbasin 1"

OBSERVATION

.....

OBSERVATION

"You arrive at cabinet 1. The episode has ended due to exceeding max steps."

AGENT_ACTION

""

Figure 10: Case-1: failure trajectories.

Failure Mode Analysis for Case-1

DOMINANT_TYPE: "MISSING_PRECONDITION"
DETAIL: "The agent picked up the soapbar and immediately attempted to place it at the target location without first checking whether the object satisfied the task precondition (clean). The task description explicitly states 'clean soapbar', but the agent ignored this constraint until the placement failed."
"CRITICAL_STEP": "Step 3, agent navigated to the target location with a dirty soapbar instead of cleaning it first at the sinkbasin."
"CORE_LESSON": "Before transporting an object to its target location, always verify that the object satisfies all task preconditions (e.g., clean, heated, cooled). If the task specifies an object state, perform the required state-change action first."
"RETRIEVAL_QUERY": "precondition check, object state, clean before place, task constraint, soapbar, sinkbasin"

Retrieved Tasks for Case-1

[4] "TASK": "Put a clean cloth in toilet.",
"RETRIEVED_REFLECTION": "Agent placed dirty cloth directly into toilet. Should have cleaned cloth at sinkbasin first. Rule: check object state precondition before final placement."
[28] "TASK": "Put a clean sponge in bathtubbasin.",
"RETRIEVED_REFLECTION": "Agent failed to clean sponge before placing in bathtubbasin. Cleaning at faucet or sinkbasin is required when task specifies 'clean' object."
[50] "TASK": "Put a clean dishsponge in cabinet.",
"RETRIEVED_REFLECTION": "Agent must clean dishsponge at sinkbasin before placing in cabinet. Always read task description for object state requirements."
INDEX 4 REASON: "Same failure pattern: agent must clean an object (cloth) before placing it at the target. Requires sinkbasin cleaning step before placement."
INDEX 28 REASON: "Identical precondition structure: object must be cleaned before placement. Agent skipped cleaning step.",
INDEX 50 REASON: "Same root cause: MISSING_PRECONDITION of 'clean' precondition. Agent transported object without state verification."

OBSERVATION (step 1)

"You are in the middle of a room. You see a desk 1, a drawer 1, a drawer 2, a garbagecan 1, a shelf 1, a shelf 2, a shelf 3. Your task is to: examine the alarmclock with the desklamp."

AGENT_ACTION (step 1)

"go to desk 1".

OBSERVATION (step 2)

"You arrive at desk 1. On the desk 1, you see a alarmclock 1, a book 1, and a pen 1."

AGENT_ACTION (step 2)

"take alarmclock 1 from desk 1"

OBSERVATION (step 3)

"You pick up the alarmclock 1 from the desk 1."

AGENT_ACTION (step 3)

"examine alarmclock 1"

OBSERVATION (step 4)

"The alarmclock 1 is a digital alarm clock. It is currently off."

AGENT_ACTION (step 4)

"go to shelf 1"

OBSERVATION (step 5)

"You arrive at shelf 1. On the shelf 1, you see a creditcard 1 and a keychain 1."

AGENT_ACTION (step 5)

"go to shelf 2"

OBSERVATION (step 6)

"You arrive at shelf 2. On the shelf 2, you see a laptop 1."

AGENT_ACTION (step 6)

"go to shelf 3"

OBSERVATION

.....

AGENT_ACTION

"The episode has ended due to exceeding max steps. Desklamp was never found or used."

AGENT_ACTION

""

Figure 11: Case-2: failure trajectories.

Failure Mode Analysis for Case-2

DOMINANT_TYPE: "WRONG_TARGET_LOCATION"
DETAIL: "The agent correctly identified the need for a desklamp but failed to find it due to an inefficient and incomplete search strategy. The agent searched shelves and drawers but never checked the desk itself for the desklamp, which is the most likely location for a desk-related item."
"CRITICAL_STEP": "Step 3, after picking up the alarmclock, the agent should have looked for the desklamp on the desk first, but instead began an unfocused search of peripheral locations.",
"CORE_LESSON": "When searching for a tool or instrument (e.g., desklamp, knife, pan), prioritize locations semantically associated with that object type before searching peripheral locations. A desklamp is most likely on a desk; a knife is most likely in a drawer near a countertop."
"RETRIEVAL_QUERY": "search strategy, desklamp, semantic location, object search, desk, unfocused exploration, tool finding"

Retrieved Tasks for Case-2

[17] "TASK": "Examine the book with the desklamp.",
"RETRIEVED_REFLECTION": "Desklamp search should start at desk. Agent spent too many steps on shelves and drawers before checking the obvious location."
[20] "TASK": "Look at mug under the desklamp.",
"SIMILARITY_REASON": "Same WRONG_TARGET_LOCATION: desklamp not found within step budget due to poor search ordering.",
[21] "TASK": "Examine the pen with the desklamp.",
"RETRIEVED_REFLECTION": "Always check the desk for desklamp first. If not on desk, check nearby shelves. Do not exhaust steps on low-probability locations."
INDEX 17: REASON: "Identical tool-finding failure: agent must locate desklamp to examine an object. Same WRONG_TARGET_LOCATION pattern.",
INDEX 20: REASON: "Agent wasted steps searching random locations for desklamp. Desklamp is almost always on the desk. Check desk first before exploring other furniture."
INDEX 21: REASON: "Same root cause: agent needs to find desklamp but uses inefficient search. Semantic location heuristic applies.",