SKILLRISE: AGENTIC REINFORCEMENT LEARNING FOR CROSS-TASK SKILL EVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model agents often encounter related yet distinct tasks that share reusable solution patterns. Yet standard agentic reinforcement learning treats tasks as independent episodes, while existing approaches to skill learning either focus on repeated attempts of one task or use pipelines with multiple stages that entangle extraction, retrieval, and execution. We introduce SkillRise, a unified reinforcement learning framework for learning skills across tasks. SkillRise organizes related instances into progressively challenging sequences and uses a single policy to alternate between task solving and curating an evolving skill document passed directly to the next task. Decoupled credit assignment across tasks supervises solving with the current task outcome and curation with discounted downstream outcomes. Experiments on ALFWorld, WebShop, and ScienceWorld show that SkillRise achieves the strongest Pass@1 performance among the compared methods, with gains over the strongest baseline ranging from 2.3 to 8.5 percentage points. Although trained across distinct tasks, its learned curation policy remains effective for repeated attempts on the same task. Further analysis reveals scaling at test time across tasks: performance improves with longer sequences of related tasks even when each task is attempted only once. This trend suggests that SkillRise reuses transferable skills across tasks rather than benefiting from repeated sampling of the same task. SkillRise further retains strong performance while substantially reducing the runtime overhead of skill learning pipelines with multiple stages. Together, these results provide a simple and efficient training paradigm for LLM agents to extract, refine, and reuse transferable skills across tasks.

1 INTRODUCTION

Large language model agents are increasingly deployed to solve complex, long-horizon tasks (Anthropic, 2025; OpenAI, 2025; Guo et al., 2025; Xie et al., 2024). In practice, they often encounter streams of related yet distinct tasks that share underlying regularities and reusable solution patterns (Zheng et al., 2025; Ouyang et al., 2026). However, standard agentic RL typically treats each task as an independent episode, discarding the experience acquired during interaction and forcing the agent to repeatedly explore from scratch (Qiu et al., 2026). An ideal agent should not only solve the task at hand, but also extract transferable skills from experience and continually reuse and refine them across subsequent tasks, becoming increasingly capable over time.

Recent work has begun to use reinforcement learning to train agents to extract reusable experience or skills from interaction trajectories (Zhang et al., 2026a; Shi et al., 2026; Wang et al., 2026a). One line repeatedly attempts the same task and summarizes previous trajectories to improve subsequent attempts (Jiang et al., 2025); however, the resulting knowledge can remain instance-specific, requires additional interactions, and is not directly optimized for cross-task transfer. Another line maintains an external experience or skill bank, updating it with newly acquired knowledge and retrieving relevant skills to guide future tasks (Xia et al., 2026; Wu et al., 2025). However, because downstream performance is jointly determined by skill extraction, retrieval, and execution, it is difficult to attribute success or failure to the quality of the extracted skills themselves.

These limitations motivate a simple, end-to-end formulation of cross-task skill learning. We consider a sequence of related yet distinct tasks, in which an agent extracts skills from earlier interactions and applies them to subsequent tasks. Within this sequence, the same policy incorporates

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.

each interaction trajectory into an evolving skill document and directly uses the document for the next task, avoiding a complex external skill-management pipeline. Since the tasks share common regularities while differing in their specific instances, performance on later tasks provides a natural measure of skill transferability and enables the policy to jointly learn to produce and use transferable skills. To optimize these two behaviors effectively, we further argue that supervision should follow their temporal roles, with task solving evaluated by the current task outcome and skill curation by performance on subsequent tasks.

Building on this formulation, we introduce SkillRise, an end-to-end reinforcement learning framework for cross-task skill learning. SkillRise first constructs progressively challenging task sequences by selecting similar yet distinct instances from the same task family and ordering them by difficulty, allowing experience from earlier tasks to support later ones. During each rollout, a single policy alternates between solving the current task with an evolving skill document and curating the document based on the resulting trajectory before proceeding to the next task instance. SkillRise employs decoupled cross-task credit assignment that assigns the current task reward to task solving and a discounted return over subsequent task rewards to skill curation. Group-relative advantages are then computed over trials sharing the same task group, sequence stage, and behavioral phase. Together, these designs enable the policy to continually extract, refine, and reuse transferable skills while solving a sequence of tasks.

We evaluate SkillRise on ALFWorld, WebShop, and ScienceWorld. Across all three agentic benchmarks, SkillRise consistently outperforms prompting-based methods and standard reinforcement learning baselines. Its advantage over methods that repeatedly attempt the same task indicates that the extracted skills transfer across different task instances. More importantly, SkillRise exhibits cross-task test-time scaling, improving as it encounters longer sequences of related tasks at test time and progressively refines its skill document. Competing methods do not show this growth trend. These results show that SkillRise learns a transferable mechanism for cross-task self-improvement, allowing skills acquired from earlier tasks to systematically improve performance on later ones.

Overall, our contributions can be summarized as follows:

- We formulate cross-task skill learning over ordered sequences of related yet distinct task instances, where later-task outcomes directly evaluate whether skills curated from earlier interactions transfer to subsequent tasks.
- We develop SkillRise, an end-to-end reinforcement learning framework that uses a single policy to solve tasks and curate an evolving skill document, together with decoupled cross-task credit assignment and role-aware group-relative optimization.
- Extensive experiments on ALFWorld, WebShop, and ScienceWorld show that SkillRise achieves the best overall results among the compared methods. Further analyses demonstrate within-task generalization and cross-task test-time scaling, and show that its compact end-to-end design offers a better balance of effectiveness and efficiency than multi-stage skill-learning pipelines.

2 PRELIMINARIES

We consider an LLM agent that interacts with an environment to solve a task instruction x sampled from a task distribution $\mathcal{D}$. At each step t, the agent receives an observation $o_t \in \mathcal{O}$ and selects an action $a_t \in \mathcal{A}$ according to

$$a_t \sim \pi_\theta(\cdot \mid x, h_t), \tag{1}$$

where π_θ denotes the agent policy and $h_t = (o_0, a_0, \ldots, a_{t-1}, o_t)$ is the interaction history available to the agent. We leave the concrete representations of observations and actions environment-dependent. Observations encode the task-relevant information available to the agent, while actions denote executable decisions that may change or query the environment. After executing a_t, the environment returns a new observation o_{t+1}, which is appended to the interaction history.

An episode produces a trajectory $\tau = (o_0, a_0, \ldots, a_{T-1}, o_T)$ and terminates when the task is completed or the interaction budget is exhausted. The complete trajectory is evaluated by a verifiable outcome reward $R(\tau)$ that measures the degree to which the task objective has been achieved. Standard agentic reinforcement learning maximizes the expected outcome over independently sampled

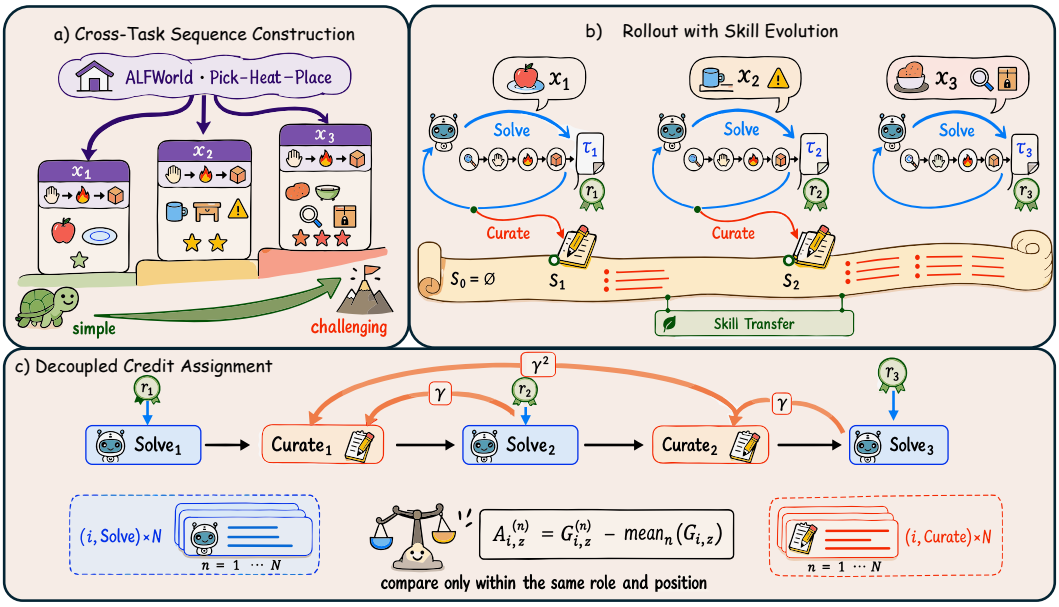

Figure 1: Overview of SkillRise. (a) Related task instances are organized from simple to challenging. (b) A shared policy alternates between solving each task and curating an evolving skill document, which serves as the sole information channel across tasks. (c) Task solving receives the current task reward, while skill curation receives discounted rewards from subsequent tasks; group-relative advantages are computed among trials with the same role and sequence position.

tasks:

$$\mathcal{J}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\,\tau\sim\pi_\theta(\cdot|x)}\left[R(\tau)\right]. \tag{2}$$

Training on multiple tasks can improve the policy through shared parameter updates, but the standard formulation still treats every interaction as an independent episode. Each new episode starts with a fresh interaction history and receives no explicit information distilled from previous episodes. Moreover, the reward $R(\tau)$ evaluates behavior only within the current episode and therefore provides no direct signal for determining whether knowledge acquired from one interaction benefits a later task. In the following section, we address these limitations by extending the independent-task formulation to ordered sequences of related tasks connected through an evolving skill document.

3 Method

Figure 1 presents an overview of SkillRise. The framework first organizes related task instances into a progressive sequence, then uses a single policy to alternate between task solving and skill curation along the sequence. The two roles are optimized with temporally aligned learning signals to jointly improve task-solving and cross-task skill-learning capabilities.

3.1 Cross-Task Sequence Construction

Standard agentic reinforcement learning treats each task as an independent episode, preventing skills learned from one task from being transferred to another. To create a learning signal for cross-task skill transfer, we organize tasks into ordered sequences

$$\mathbf{x} = (x_1, \ldots, x_K) \sim \mathcal{G}, \tag{3}$$

where $\mathcal{G}$ is a distribution over sequences of K distinct instances from the same task family. Although these instances differ in their concrete entities and goals, they share common interaction routines that can be abstracted into transferable skills. Consequently, skills learned from earlier tasks can guide later tasks, while performance on the later tasks provides a direct measure of whether those skills transfer beyond a single instance.

To construct the sequences, we first partition tasks by environment-provided family metadata, and then greedily group diverse instances within each family. We arrange the selected tasks from relatively simple to more complex according to their task attributes. For example, we group WebShop tasks by product category and order distinct requests by the number of required attributes and options. The resulting tasks are executed sequentially and connected through an evolving skill document.

3.2 Cross-Task Rollout with Skill Evolution

Given a cross-task sequence $\mathbf{x}$, SkillRise performs a sequential rollout while maintaining a single textual skill document S_i that summarizes the transferable skills acquired through the first i tasks. Each trial starts from an empty document $S_0 = \varnothing$. For the i-th task, the policy first uses the current document to generate a task-solving trajectory and receive its task reward:

$$\tau_i \sim \pi_\theta(\cdot \mid x_i, S_{i-1}), \qquad r_i = R(\tau_i). \tag{4}$$

For every task except the last, the same policy then switches from task solving to skill curation and produces a revised document:

$$S_i \sim \pi_\theta(\cdot \mid S_{i-1}, \tau_i, r_i), \qquad i < K. \tag{5}$$

The two phases share the same policy parameters and differ only in their role-specific instructions.

During curation, the policy rewrites the complete document: it preserves useful skills from S_{i-1}, consolidates successful procedures or failure modes revealed by τ_i, and removes instance-specific details. Only the revised document S_i is provided to task x_{i+1}; earlier task trajectories are not carried into its solving context. Hence, S_i is the sole information channel across tasks, and its utility is reflected by performance on subsequent tasks. During training, we sample N independent trials for each sequence, with every trial starting from an empty document and evolving its own skill document.

3.3 Decoupled Credit Assignment and Optimization

Decoupled credit assignment. Task solving and skill curation play different temporal roles. A solving trajectory can be evaluated directly by the current task reward, whereas the revised skill document is produced only after the task terminates and can influence only subsequent tasks. Assigning the same sequence-level return to both phases would therefore conflate task-solving quality with skill transferability. Let $n \in \{1, \ldots, N\}$ index the independent trials and $z \in \{\mathrm{solve}, \mathrm{curate}\}$ denote the phase. We assign the phase-level return

$$G_{i,z}^{(n)} = \begin{cases} r_i^{(n)}, & z = \mathrm{solve}, \\ \displaystyle\sum_{j=i+1}^{K} \gamma^{j-i} r_j^{(n)}, & z = \mathrm{curate},\ i < K, \end{cases} \tag{6}$$

where $\gamma \in [0, 1]$ is the cross-task discount factor. Thus, solving is supervised only by the current task outcome, while curation is evaluated by the discounted outcomes of later tasks. The discount places more credit on nearby tasks, whose performance is more directly affected by the revised document.

Role-aware group-relative optimization. For a fixed sequence, task position i, and phase z, we compare the returns of its N trials:

$$\overline{G}_{i,z} = \frac{1}{N} \sum_{n=1}^{N} G_{i,z}^{(n)}, \qquad A_{i,z}^{(n)} = G_{i,z}^{(n)} - \overline{G}_{i,z}. \tag{7}$$

Grouping by both task position and phase ensures that solving and curation returns never serve as baselines for one another, while also accounting for differences across positions in the sequence.

Let $\rho_{i,z,t}^{(n)}(\theta)$ denote the token-level importance ratio between the current and old policies for token t. We jointly optimize all solving and curation responses using the clipped objective

$$\mathcal{J}(\theta) = \mathbb{E}_{n,i,z,t}\left[\min\left(\rho_{i,z,t}^{(n)}(\theta) A_{i,z}^{(n)}, \mathrm{clip}\Big(\rho_{i,z,t}^{(n)}(\theta), 1-\epsilon, 1+\epsilon\Big) A_{i,z}^{(n)}\right)\right]. \tag{8}$$

Table 1: Pass@1 success rate (%) of Qwen3-4B on ALFWorld, WebShop, and ScienceWorld. Best results are shown in bold and second-best results are underlined.

	ALFWorld							WebShop	ScienceWorld
Method	Pick	Look	Clean	Heat	Cool	Pick2	Avg.	SR	SR
Zero-shot	1.5	6.1	3.6	4.2	2.8	8.6	4.4	1.4	0.8
ReAct	45.5	30.3	21.4	12.5	18.1	23.5	25.0	2.1	7.8
Reflexion	74.2	39.4	19.0	25.0	13.9	13.6	28.9	2.7	7.8
PPO	82.1	70.0	57.9	52.9	50.0	**72.0**	67.2	74.2	39.1
RLOO	88.2	37.5	96.1	61.9	<u>80.0</u>	42.1	74.2	76.6	43.0
GRPO	<u>94.1</u>	75.0	<u>96.2</u>	66.7	75.0	52.6	79.7	75.8	42.2
GiGPO	90.3	**88.2**	**96.4**	**83.3**	75.0	60.0	<u>83.6</u>	<u>77.3</u>	<u>46.1</u>
LaMer	88.2	62.5	<u>96.2</u>	71.4	75.0	42.1	76.7	74.2	41.4
SkillRise	**100.0**	<u>87.5</u>	88.5	<u>81.0</u>	**80.8**	<u>68.4</u>	**85.9**	**84.4**	**54.6**

where ϵ is the clipping range. The phase-level advantage is applied to all policy-generated tokens in the corresponding phase. Since the two phases share the same policy, this objective jointly improves task solving and skill curation while keeping their learning signals separated.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate on three interactive text environments. ALFWorld (Shridhar et al., 2020) requires an embodied household agent to navigate and manipulate objects; we use its TextWorld version and report results for all six task families: Pick, Look, Clean, Heat, Cool, and Pick2. WebShop (Yao et al., 2022a) requires an agent to search for, configure, and purchase a product that satisfies a natural-language request. We use the 1,000-product version with synthetic goals. ScienceWorld (Wang et al., 2022) evaluates grounded scientific reasoning through interactive experiments. We evaluate on 128 held-out task instances from each environment.

Baselines. We compare against three prompting baselines: Zero-shot, ReAct (Yao et al., 2022b), and Reflexion (Shinn et al., 2023). We further include four task-independent RL algorithms: PPO (Schulman et al., 2017), RLOO (Ahmadian et al., 2024), GRPO (Shao et al., 2024), and GiGPO (Feng et al., 2025). Finally, we compare with LaMer (Jiang et al., 2025), a Meta-RL baseline that adapts across repeated attempts of the same task through textual reflection.

Implementation Details. All methods use Qwen3-1.7/4B (Yang et al., 2025) as the backbone. For SkillRise, each training batch contains 16 sequences, each with $K = 3$ tasks and $N = 8$ independent trials. LaMer also uses eight trials with three attempts, whereas the task-independent RL baselines use 24 rollouts per task. Thus, every trainable method processes the same 384 task plays per update. We use an actor learning rate of 1×10^{-6}, a mini-batch size of 128, a maximum response length of 1,024 tokens, and train for up to 150 updates. For SkillRise, the cross-task discount factor is set to $\gamma = 0.6$. Evaluation uses sampling with temperature 0.7. Training is implemented with FSDP and vLLM, and all experiments are conducted on eight NVIDIA H800 GPUs.

4.2 Main Results

Overall Performance. As shown in Table 1, SkillRise achieves the strongest overall Pass@1 performance across all three benchmarks. It obtains 85.9% on ALFWorld, 84.4% on WebShop, and 54.6% on ScienceWorld, outperforming the strongest baseline, GiGPO, by 2.3, 7.1, and 8.5 percentage points, respectively. On ALFWorld, SkillRise ranks first or second in five of the six task families, leading to the best overall success rate despite substantial variation across task types. We attribute these gains to cross-task training, where related but distinct instances repeatedly expose the policy to different realizations of the same underlying interaction routines. This structured training

Table 2: Pass@1/2/3 success rates (%) of Qwen3-4B on ALFWorld, WebShop, and ScienceWorld. Best results are shown in bold and second-best results are underlined.

	ALFWorld			WebShop			ScienceWorld		
Method	Pass@1	Pass@2	Pass@3	Pass@1	Pass@2	Pass@3	Pass@1	Pass@2	Pass@3
Zero-shot	4.4	9.4	12.5	1.4	2.1	2.3	0.8	1.6	2.3
ReAct	25.0	45.3	57.0	2.1	4.5	4.5	7.8	10.2	12.5
Reflexion	28.9	36.7	42.2	2.7	3.3	3.5	7.8	12.5	17.2
PPO	67.2	69.5	73.4	74.2	75.8	75.8	39.1	45.3	47.7
RLOO	74.2	80.5	82.8	76.6	78.1	78.9	43.0	48.4	52.3
GRPO	79.7	81.3	82.8	75.8	76.6	78.1	42.2	50.0	50.0
GiGPO	83.6	86.7	89.1	77.3	79.7	80.5	46.1	50.8	55.5
LaMer	76.7	86.7	91.4	74.2	87.5	94.5	41.4	46.1	46.8
SkillRise	**85.9**	**91.4**	**92.2**	**84.4**	**93.8**	**96.1**	**54.6**	**58.5**	**61.0**

signal makes transferable regularities easier to identify and reinforce across tasks, accelerating their internalization into the model parameters and enabling the policy to acquire reusable task-solving skills more efficiently.

Generalization to Within-Task Adaptation. Although SkillRise learns to curate skills from sequences of distinct tasks during training, we evaluate Pass@2 and Pass@3 by retrying the same held-out task while carrying forward the evolving skill document. This protocol tests whether the skill-learning strategy acquired across tasks can generalize to within-task adaptation. As shown in Table 2, SkillRise achieves the best Pass@2 and Pass@3 on all three benchmarks. At Pass@3, it reaches 92.2%, 96.1%, and 61.0% on ALFWorld, WebShop, and ScienceWorld, exceeding the strongest baseline by 0.8, 1.6, and 5.5 percentage points, respectively. Notably, it also outperforms LaMer, which is trained explicitly on repeated attempts of the same task, by 0.8, 1.6, and 14.2 points. These results suggest that the learned curation policy captures a general strategy for converting interaction outcomes into actionable skills, which transfers beyond its cross-task training setting to guide subsequent attempts on the same task.

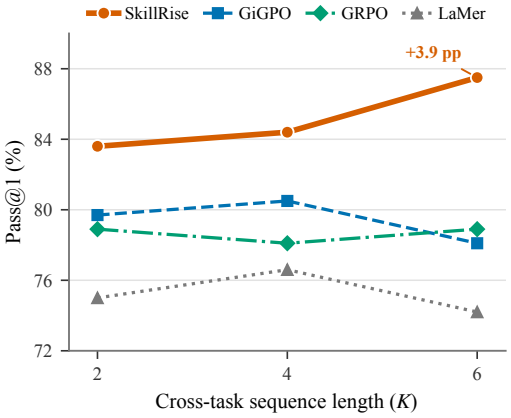

Figure 2: Cross-task test-time scaling on ALFWorld. The same 128 held-out tasks are partitioned into related sequences of length K, with each task attempted once.

5 Analysis

5.1 Cross-Task Test-Time Scaling

We investigate whether the cross-task skill-learning capability acquired during training scales with the amount of related task context available at test time. We partition the same held-out task set into sequences of $K \in \{2, 4, 6\}$ tasks and allow SkillRise to carry forward and update its skill document within each sequence. As shown in Figure 2, SkillRise improves monotonically over the evaluated sequence lengths, from 83.6% at $K = 2$ to 87.5% at $K = 6$. In contrast, LaMer, GiGPO, and GRPO show no sustained improvement as K grows, and the margin over the strongest baseline widens from 3.9 to 8.6 percentage points. Since every task receives only one attempt, this gain does not result from additional sampling on the same instance. Instead, the trend suggests that the trained policy can transform interactions with earlier tasks into transferable skills for later tasks, with the benefit becoming more pronounced as the skill document is progressively refined over longer sequences.

Table 3: Pass@1 success rate (%) on ALFWorld across Qwen3 model sizes. Δ denotes the improvement from 1.7B to 4B. Best results are shown in bold.

	Qwen3		
Method	1.7B	4B	Δ
GRPO	75.0	79.7	$+4.7$
LaMer	73.4	76.7	$+3.3$
SkillRise	**78.1**	**85.9**	$+\mathbf{7.8}$

5.2 Performance across Model Sizes

We further evaluate whether the gains of SkillRise extend beyond a single model size. As shown in Table 3, SkillRise achieves the strongest Pass@1 performance at both evaluated scales. With Qwen3-1.7B, SkillRise reaches 78.1%, exceeding the strongest baseline by 3.1 percentage points. With Qwen3-4B, it further improves to 85.9%, widening this margin to 6.2 points. Thus, the advantage of cross-task skill learning is already present with the smaller backbone and becomes more pronounced at the larger scale.

All three methods benefit from increasing the backbone size, but the magnitude of improvement differs substantially. SkillRise gains 7.8 points from 1.7B to 4B, compared with 4.7 points for GRPO and 3.3 points for LaMer. Unlike task-independent optimization, SkillRise must jointly learn to solve the current task, identify regularities shared across related tasks, and curate them into a reusable skill document. The larger gain suggests that additional model capacity is particularly useful for this combination of task execution and cross-task abstraction. Overall, these results show that the effectiveness of SkillRise is not confined to a single parameter scale and that, within the evaluated range, its advantage increases as the backbone grows.

5.3 Ablation Studies

Sensitivity to Cross-Task Discounting. We vary the cross-task discount factor over $\gamma \in \{0.3, 0.4, 0.6, 0.7\}$ while keeping the remaining settings fixed. As shown in the left panel of Figure 3, all four variants follow closely aligned learning trajectories, exhibiting comparable improvements throughout training and converging within approximately one percentage point of one another. This consistency shows that the gains of SkillRise do not rely on carefully tuning the relative weight assigned to future tasks. Instead, the cross-task learning signal remains effective over a broad range of discount factors, indicating that the proposed credit assignment is robust to reasonable variations in how strongly downstream outcomes supervise skill curation.

Effect of Skill Curation. The no-curation variant retains the same $K = 3$ task sequences and environment interaction budget, but removes the inter-task curation phase and prevents skill transfer between tasks. As shown in the right panel of Figure 3, the three methods improve similarly during the early stage, but SkillRise begins to separate after sufficient cross-task interactions have accumulated and maintains the strongest training score thereafter. At the end of training, it leads the no-curation variant by roughly three percentage points and GRPO by more than six points. The advantage over the no-curation variant shows that training on related task sequences alone is insufficient. Explicitly transforming completed trajectories into an evolving skill document allows useful procedures discovered on earlier tasks to be consolidated and reused on later ones. The additional improvement over task-independent GRPO further suggests that cross-task skill accumulation provides a complementary learning signal beyond optimizing each task in isolation.

Overall, these ablations confirm the importance of skill curation while showing that SkillRise remains robust to the choice of cross-task discount factor. Together, they support the effectiveness of the complete framework.

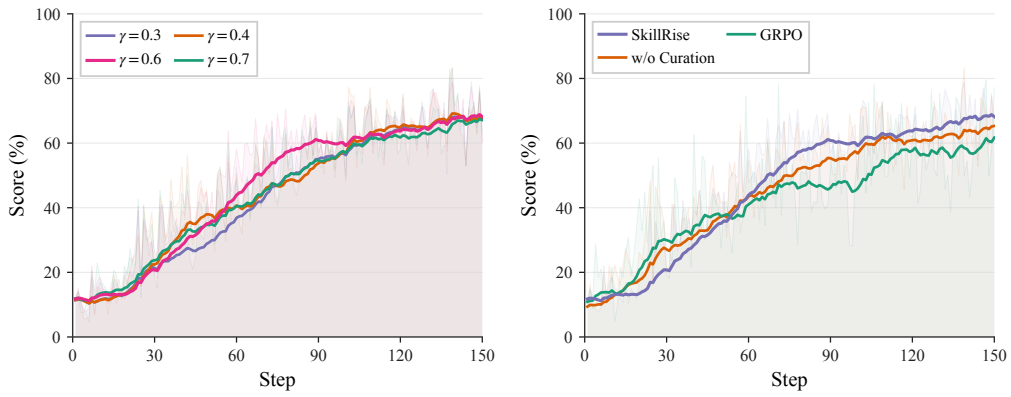

Figure 3: Training dynamics on ALFWorld in terms of training reward (%). Thin lines show the raw rewards, while thick lines show the smoothed rewards. Left: sensitivity to the cross-task discount factor. Right: comparison of SkillRise, its no-curation variant, and GRPO.

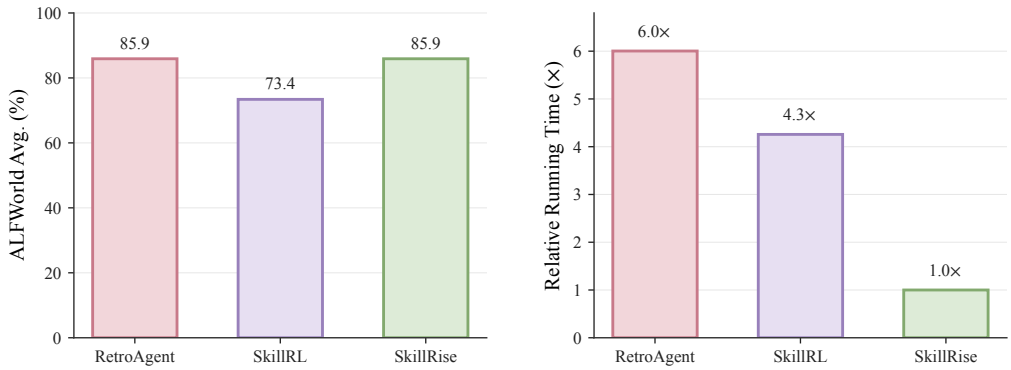

Figure 4: Comparison of skill-learning pipelines on ALFWorld. Left: average success rate. Right: relative running time normalized by SkillRise.

5.4 Pipeline Complexity and Training Efficiency

Existing skill-learning methods often decompose skill acquisition into a multi-stage management pipeline. RetroAgent generates reflections, stores them in a growing memory buffer, retrieves relevant lessons, and updates their utilities (Zhang et al., 2026b). SkillRL further relies on teacher-based trajectory distillation, hierarchical SkillBank construction, cold-start SFT, skill retrieval, and iterative bank refinement (Xia et al., 2026). For our SkillRL reproduction, we use Gemini-2.5-Pro as the teacher model. In such pipelines, downstream performance is jointly determined by skill extraction, storage, retrieval, updating, and execution. This coupling makes it difficult to attribute success or failure to the learned skill itself, while the additional stages also introduce substantial computational overhead.

SkillRise instead adopts a compact end-to-end training paradigm: the same policy solves each task, curates a single sequence-local skill document, and directly uses it on the next task. This removes the need for an external teacher, separate memory maintenance, and an additional retrieval module. As shown in Figure 4, SkillRise achieves an 85.9% average success rate on ALFWorld, matching RetroAgent and outperforming SkillRL by 12.5 percentage points. Under the same GPU configuration, RetroAgent and SkillRL require $6.0\times$ and $4.3\times$ the end-to-end running time of SkillRise, respectively. Thus, transferable skill learning does not require a long external management pipeline. Directly supervising skill curation through subsequent task outcomes provides a clearer learning signal while retaining strong performance with substantially lower training overhead.

6 Related Work

Reinforcement Learning for LLM Agents. Reinforcement learning has become a central approach for improving LLMs, evolving from preference alignment with human feedback to reinforcement learning with verifiable rewards, which provides scalable supervision for reasoning and improves training efficiency (Shao et al., 2024; Guo et al., 2025; Yao et al., 2026; Chen et al., 2026). As LLMs are increasingly deployed as interactive agents, agentic RL extends this paradigm from single-turn generation to multi-turn decision making, where long horizons and delayed outcomes make exploration and credit assignment substantially harder (Zhang et al., 2025). Existing methods develop hierarchical, step-level, or milestone-guided credit assignment (Feng et al., 2025; Zhang et al., 2026c; He et al., 2026; Wang et al., 2026b; Lu et al., 2026a), improve exploration through autonomous environment interaction and online curricula, including for web navigation and long-context software engineering (Ye et al., 2026; Qi et al., 2025; Wei et al., 2025; Wang et al., 2025), or provide scalable training abstractions for heterogeneous agent workflows (Luo et al., 2025; Xi et al., 2025). These advances demonstrate that outcome-based supervision can improve long-horizon interaction, but most methods still optimize independently sampled task episodes. Consequently, how an agent can learn from the relationship among episodes and turn earlier interactions into reusable procedural knowledge remains less explored.

Experience and Skill Learning in LLM Agents. Learning from past interactions is a central mechanism for self-evolving agents, with the core challenge being how to transform interaction histories into reusable knowledge (Gao et al., 2025). Concrete representations retain trajectories, examples, or verbal reflections for later reuse (Shinn et al., 2023; Zhao et al., 2024). More abstract representations distill histories into guidelines, workflows, or reasoning strategies that can generalize across instances (Fu et al., 2024; Wang et al., 2024; Ouyang et al., 2025). Recently, editable skill artifacts have emerged as a modular form of procedural memory, allowing agents to consolidate, revise, and reuse high-level solution procedures (Ni et al., 2026; Yang et al., 2026). Reinforcement learning is increasingly used to train different stages of this skill lifecycle, including skill utilization, internalization, extraction, and curation (Xia et al., 2026; Lu et al., 2026b; Ouyang et al., 2026). LaMer formulates adaptation as a cross-episode Meta-RL problem: the agent repeatedly interacts with the same task and conditions later attempts on environment feedback and textual reflections (Jiang et al., 2025). For skill evolution, Evolving-RL jointly optimizes skill extraction and skill-conditioned solving by evaluating extracted skills on related downstream tasks (Fan et al., 2026). SkillOS keeps the executor frozen and trains a separate curator to update a skill repository, using later performance over related task streams as delayed supervision (Ouyang et al., 2026). These methods manage skill extraction, storage, retrieval, and execution in different ways. However, multi-stage pipelines can make it difficult to attribute downstream performance to individual components.

7 Conclusion

We introduced SkillRise, an end-to-end reinforcement learning framework for cross-task skill learning. It organizes related yet distinct tasks into ordered sequences, where a single policy alternates between task solving and curating an evolving skill document. Because this document serves as the sole information channel across tasks, performance on subsequent tasks directly reflects whether the knowledge extracted from earlier interactions is transferable. Decoupled credit assignment further aligns supervision with the temporal roles of the two behaviors, using current-task outcomes for task solving and discounted downstream outcomes for skill curation. Across ALFWorld, WebShop, and ScienceWorld, SkillRise achieves the strongest overall Pass@1 performance among the compared methods and generalizes from cross-task training to repeated attempts of the same held-out task. On ALFWorld, its performance further improves with longer sequences of related tasks at test time, even though each task is attempted only once, indicating that the gains come from accumulating transferable knowledge rather than repeatedly sampling the same task. Moreover, SkillRise matches or exceeds representative multi-stage skill-learning pipelines on ALFWorld while reducing end-to-end running time by up to $6.0\times$. Together, these results demonstrate that LLM agents can improve at test time by continually extracting, refining, and reusing transferable skills across related tasks, providing a simple and efficient training paradigm for developing agents that become more capable as they accumulate experience.

AI USE STATEMENT

In preparing this manuscript, we used generative AI tools solely for language polishing, including correcting grammatical errors, refining wording, and improving the clarity and readability of text written by the authors. These tools were not used to develop research ideas or design methods or experiments. The authors reviewed all AI-assisted edits to ensure that they preserved the intended meaning and technical content. The authors take full responsibility for the accuracy, originality, and integrity of the final manuscript.

ETHICS STATEMENT

This work uses publicly available agent benchmarks (ALFWorld, WebShop, and ScienceWorld) and open-weight Qwen models, and does not involve human participants or the collection of personal data. Training and evaluation are conducted in simulated text-based environments. In particular, WebShop uses synthetic goals, and its purchase actions do not initiate real-world transactions. Deploying agents trained with SkillRise in real-world environments should include appropriate safety validation, permission controls, and human oversight. The underlying models and benchmarks should be used in accordance with their licenses and usage restrictions.

REFERENCES

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.

Anthropic. Claude sonnet 4.5 model card. 2025. URL https://www.anthropic.com/news/claude-sonnet-4-5.

Yuxin Chen, Yu Wang, Yi Zhang, Ziang Ye, Zhengzhou Cai, Yaorui Shi, Qi Gu, Hui Su, Xunliang Cai, Xiang Wang, et al. Learning to self-verify makes language models better reasoners. *arXiv preprint arXiv:2602.07594*, 2026.

Zhiyuan Fan, Wenwei Jin, Feng Zhang, Bin Li, Yihong Dong, Yao Hu, and Jiawei Li. Evolving-rl: End-to-end optimization of experience-driven self-evolving capability within agents. *arXiv preprint arXiv:2605.10663*, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training, 2025. URL https://arxiv.org/abs/2505.10978.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents. *Advances in Neural Information Processing Systems*, 37: 119919–119948, 2024.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. 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*, 2025.

Daya Guo et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Shuo He, Lang Feng, Qi Wei, Xin Cheng, Lei Feng, and Bo An. Hierarchy-of-groups policy optimization for long-horizon agentic tasks. *arXiv preprint arXiv:2602.22817*, 2026.

Yulun Jiang, Liangze Jiang, Damien Teney, Michael Moor, and Maria Brbic. Meta-rl induces exploration in language agents. *arXiv preprint arXiv:2512.16848*, 2025.

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

Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Skill0: In-context agentic reinforcement learning for skill internalization. *arXiv preprint arXiv:2604.02268*, 2026b.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K Qiu, and Yuqing Yang. Agent lightning: Train any ai agents with reinforcement learning. *arXiv preprint arXiv:2508.03680*, 2025.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2skill: Distill trajectory-local lessons into transferable agent skills. *arXiv preprint arXiv:2603.25158*, 2026.

OpenAI. Introducing gpt-5. 2025. URL `https://openai.com/index/introducing-gpt-5/`.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*, 2025.

Siru Ouyang, Jun Yan, Yanfei Chen, Rujun Han, Zifeng Wang, Bhavana Dalvi Mishra, Rui Meng, Chun-Liang Li, Yizhu Jiao, Kaiwen Zha, et al. Skillos: Learning skill curation for self-evolving agents. *arXiv preprint arXiv:2605.06614*, 2026.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, et al. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 79791–79821, 2025.

Libin Qiu, Zhirong Gao, Junfu Chen, Yuhang Ye, Weizhi Huang, Xiaobo Xue, Wenkai Qiu, and Shuo Tang. Autorefine: From trajectories to reusable expertise for continual llm agent refinement. *arXiv preprint arXiv:2601.22758*, 2026.

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, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yaorui Shi, Yuxin Chen, Zhengxi Lu, Yuchun Miao, Shugui Liu, Qi Gu, Xunliang Cai, Xiang Wang, and An Zhang. Skill1: Unified evolution of skill-augmented agents via reinforcement learning. *arXiv preprint arXiv:2605.06130*, 2026.

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

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

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1529–1550, 2026a.

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

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

Zixuan Wang, Yuchen Yan, Hongxing Li, Teng Pan, Dingming Li, Ruiqing Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Milestone-guided policy learning for long-horizon language agents. *arXiv preprint arXiv:2605.06078*, 2026b.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I. Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution, 2025. URL https://arxiv.org/abs/2502.18449.

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

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Honglin Guo, Jiaqi Liu, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, et al. Agentgym-rl: Training llm agents for long-horizon decision making through multi-turn reinforcement learning. *arXiv preprint arXiv:2509.08755*, 2025.

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

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.

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

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, et al. Skillopt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022b.

Zhiyuan Yao, Yi-Kai Zhang, Yuxin Chen, Yueqing Sun, Zishan Xu, Yu Yang, Tianhao Hu, Qi Gu, Hui Su, and Xunliang Cai. Coba-rl: Capability-oriented budget allocation for reinforcement learning in llms. *arXiv preprint arXiv:2602.03048*, 2026.

Ziang Ye, Wentao Shi, Yuxin Liu, Yu Wang, Zhengzhou Cai, Yaorui Shi, Qi Gu, Xunliang Cai, and Fuli Feng. Look before you leap: Autonomous exploration for llm agents. *arXiv preprint arXiv:2605.16143*, 2026.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, et al. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*, 2025.

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

Xiaoying Zhang, Zichen Liu, Yipeng Zhang, Xia Hu, and Wenqi Shao. Retroagent: From solving to evolving via retrospective dual intrinsic feedback. *arXiv preprint arXiv:2603.08561*, 2026b.

Zhen Zhang, Kaiqiang Song, Xun Wang, Yebowen Hu, Weixiang Yan, et al. Cm2: Reinforcement learning with checklist rewards for multi-turn and multi-step agentic tool use, 2026c. URL https://arxiv.org/abs/2602.12268.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, ZhongZhi Li, Yingying Zhang, Le Song, and Qianli Ma. Lifelongagentbench: Evaluating llm agents as lifelong learners. *arXiv preprint arXiv:2505.11942*, 2025.

A LIMITATION

Our current formulation assumes access to task-family metadata for constructing sequences of related instances; automatically discovering such relationships in open-ended task streams remains an important direction. Moreover, due to computational constraints, our experiments are limited to models of up to 4B parameters, leaving evaluation at larger model scales for future work. Finally, our experiments focus on three text-based agent benchmarks and verifiable outcome rewards. Evaluating the framework on broader multimodal, real-world, and less readily verifiable tasks would further establish its generality.

B PROMPTS

We reproduce the user-message templates used for task solving and skill curation. Braced fields are filled at runtime. At the start of each task sequence, an empty skill document is represented by `(The skill document is currently empty. No skills have been distilled yet.)`

Task-Solving Prompt of SkillRise on ALFWorld

```
You are an expert agent operating in the ALFRED Embodied Environment.
## Current Skill Document
The following skills were distilled from earlier tasks of the SAME family. Use them only
when you are confident they apply to the current situation; ignore any skill that does not
fit.
{skill_document}
{init_observation}{current_trajectory}
Your admissible actions of the current situation are:
[{admissible_actions}]
Now it's your turn to take an action.

      • Your response should first be step-by-step reasoning about the current situation.
      • Once you've finished your reasoning, you should choose an admissible action for
        the current step and present it within <action> </action> tags.
```

Figure 5: SkillRise task-solving prompt for ALFWorld.

Task-Solving Prompt of SkillRise on WebShop

```
You are an expert autonomous agent operating in the WebShop e-commerce environment.
## Current Skill Document
The following skills were distilled from earlier shopping tasks of the SAME family. Use
them only when you are confident they apply to the current situation; ignore any skill
that does not fit.
{skill_document}
Your task is to: {task_description}.{current_trajectory}
Your admissible actions of the current situation are:
[
{admissible_actions}
].
Now it's your turn to take one action for the current step.
Your response should first be step-by-step reasoning about the current situation, then
think carefully which admissible action best advances the shopping goal.
Once you've finished your reasoning, you should choose an admissible action for current
step and present it within <action> </action> tags.
```

Figure 6: SkillRise task-solving prompt for WebShop.

Task-Solving Prompt of SkillRise on ScienceWorld

You are an expert autonomous agent operating in the ScienceWorld environment, which is a text-based virtual environment centered around accomplishing tasks from the elementary science curriculum.

Current Skill Document

The following skills were distilled from earlier tasks of the SAME family. Use them only when you are confident they apply to the current situation; ignore any skill that does not fit.

{skill_document}

Your task is to: {task_description}.

Your current observation is: {current_observation}.{current_trajectory}

Your admissible actions of the current situation are:

[

{available_actions}

].

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 task goal.

Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

Figure 7: SkillRise task-solving prompt for ScienceWorld.

Skill-Curation Prompt of SkillRise on ALFWorld

You are maintaining a SKILL DOCUMENT for an agent solving a family of related ALFRED household tasks. You have just observed one task attempt. Your job is to revise the skill document so that it helps the agent solve the LATER tasks in this same family more reliably.

Old Skill Document

{skill_document}

Most Recent Task Attempt (observations and the actions the agent took)

{current_trajectory}

Outcome of this attempt: {outcome}

Your job

Rewrite the skill document. Start from the old document above and refine it with what this attempt revealed. Follow these rules strictly:

- ABSTRACT, do not memorize. Remove task-specific instances: replace concrete object names, receptacle ids, and location numbers (e.g., ``soapbottle 1'', ``drawer 3'') with general concepts (``the target object'', ``a receptacle'').
- Ground every statement in what actually happened in the attempt above. Do not invent steps you did not observe.
- If the attempt failed, extract the failure point as a concrete pitfall.
- Be CONCISE. Do not pad with generic advice or vague tips. A short, precise document beats a long one. If this attempt revealed little new, keep the old document almost unchanged.
- Do NOT copy the trajectory verbatim. Distill reusable procedure, not history.

Suggested (not mandatory) structure---use the headers that fit:

When to use

<high-level situations this skill family applies to>

Workflow

<ordered high-level steps that generalize across instances>

Pitfalls

<failure modes and when to deviate from the default workflow>

Write your reasoning first, then output the FULL revised skill document inside <skill> </skill> tags.

Figure 8: SkillRise skill-curation prompt for ALFWorld.

Skill-Curation Prompt of SkillRise on WebShop

You are maintaining a SKILL DOCUMENT for an agent solving a family of related WebShop online-shopping tasks. You have just observed one task attempt. Your job is to revise the skill document so that it helps the agent solve the LATER tasks in this same family (same product category) more reliably.

Old Skill Document

{skill_document}

Most Recent Task Attempt (the shopping goal, observations, and the actions the agent took)

{current_trajectory}

Outcome of this attempt: {outcome}

Your job

Rewrite the skill document. Start from the old document above and refine it with what this attempt revealed. Follow these rules strictly:

- ABSTRACT, do not memorize. Remove task-specific instances: replace concrete product names, brands, exact attribute values, and price thresholds (e.g., ``rose red'', ``1.6 ounce'', ``lower than 60.00 dollars'') with general concepts (``the required color/size option'', ``the price ceiling'').
- Ground every statement in what actually happened in the attempt above. Do not invent steps you did not observe.
- If the attempt failed, extract the failure point as a concrete pitfall.
- Be CONCISE. Do not pad with generic advice or vague tips. A short, precise document beats a long one. If this attempt revealed little new, keep the old document almost unchanged.
- Do NOT copy the trajectory verbatim. Distill reusable procedure, not history.

When useful, the document should capture WebShop-specific procedure:

- how to formulate a search query that encodes the goal's constraints,
- which search result to click (matching category/attributes, not just the first),
- the order to configure product options (color, size, count, etc.) before buying,
- verifying the product meets the required attributes AND the price ceiling BEFORE click[buy now],
- navigation patterns (when to go back, view options/description/features, paginate).

Suggested (not mandatory) structure -- use the headers that fit:

When to use

<high-level situations this skill family applies to>

Workflow

<ordered high-level steps that generalize across product instances>

Pitfalls

<failure modes and when to deviate from the default workflow>

Write your reasoning first, then output the FULL revised skill document inside <skill> </skill> tags.

Figure 9: SkillRise skill-curation prompt for WebShop.

Skill-Curation Prompt of SkillRise on ScienceWorld

You are maintaining a SKILL DOCUMENT for an agent solving a family of related ScienceWorld tasks. You have just observed one task attempt. Your job is to revise the skill document so that it helps the agent solve the LATER tasks in this same family more reliably.

Old Skill Document

{skill_document}

Most Recent Task Attempt (observations and the actions the agent took)

{current_trajectory}

Outcome of this attempt: {outcome}

Your job

Rewrite the skill document. Start from the old document above and refine it with what this attempt revealed. Follow these rules strictly:

- ABSTRACT, do not memorize. Replace concrete instances (specific substance names, container colors, room names, numeric thresholds) with general concepts (``the target substance'', ``the designated container'').
- Ground every statement in what actually happened in the attempt above. Do not invent steps you did not observe.
- If the attempt failed, extract the failure point as a concrete pitfall.
- Be CONCISE. A short, precise document beats a long one. If this attempt revealed little new, keep the old document almost unchanged.
- Do NOT copy the trajectory verbatim. Distill reusable procedure, not history.

Suggested (not mandatory) structure---use the headers that fit:

When to use

<high-level situations this skill family applies to>

Workflow

<ordered high-level steps that generalize across instances>

Pitfalls

<failure modes and when to deviate from the default workflow>

Write your reasoning first, then output the FULL revised skill document inside <skill> </skill> tags.

Figure 10: SkillRise skill-curation prompt for ScienceWorld.

Task 1 — put two spraybottle in cabinet (× failed; selected actions)

```
go to countertop 1
take spraybottle 1 from countertop 1
go to cabinet 1
open cabinet 1
move spraybottle 1 to cabinet 1          # first instance placed
go to shelf 1
go to cabinet 2
go to countertop 1
(none)                                      # no valid action
go to shelf 1
```

↓ *the observed failure is summarized in skill v1*

Curate — relevant excerpt of skill v1 after Task 1

Workflow (excerpt). (1) Go to the location where the target object is currently situated. (2) Take the target object from the location. (3) Go to the receptacle where the object should be placed. (4) Open the receptacle if it is closed. (5) Move the target object to the receptacle. **(6) If multiple instances of the target object are present, repeat steps 1–5 for each instance.** ← *new general rule*

↓ *skill v1 is included in the next solve prompt*

Task 2 — put two peppershaker in cabinet (✓ succeeded; selected actions)

```
go to diningtable 1
go to diningtable 2
take peppershaker 1 from diningtable 2
go to cabinet 1
move peppershaker 1 to cabinet 1          # first instance
go to diningtable 2
take peppershaker 2 from diningtable 2
go to cabinet 1
move peppershaker 2 to cabinet 1          # second instance; success
```

Figure 11: **Case 1: a rule recorded after failure.** After a failure in which only one of two required objects is placed, the curation output records a repeat-for-each-instance rule. The following task is conditioned on this document and successfully places both instances of a different object.

C Case Study

We show selected actions and abridged curation outputs from the first two tasks of two ALFWorld training rollouts with sequence length $K = 3$. Displayed tasks are numbered from 1. Each rollout begins with an empty skill document; between tasks, only the updated document is carried forward, not earlier-task trajectories. Ellipses mark omitted actions; arrows and comments after # are author annotations. Case 1 (Figure 11) pairs a failure-derived rule with the next task conditioned on that rule. Case 2 (Figure 12) shows an incremental document revision. These examples illustrate the update process rather than isolate a causal effect.

Case 1: distilling a lesson from failure. The first task asks for two target instances, but the agent places only one and fails. The subsequent curation output adds a general instruction to repeat the procedure for each instance. The next task receives this document, places both instances of a different object, and succeeds.

- **Task 1**: put **two** spraybottle in cabinet (× failed)
- **Task 2**: put **two** peppershaker in cabinet (✓ succeeded)

Task 1 — put some cellphone on bed (✓ succeeded; selected actions)

```
go to diningtable 1
go to armchair 1
take cellphone 1 from armchair 1
go to bed 1
move cellphone 1 to bed 1                    # no open action required;
success
```

↓ *a basic placement workflow is recorded in skill v1*

Curate — relevant excerpt of skill v1 after Task 1

Workflow (excerpt). (1) Navigate to the location where the target object is currently located. (2) Identify the target object on the surface of the receptacle. (3) Pick up the target object from the receptacle. (4) Navigate to the destination location. (5) Place the target object on the destination receptacle.

↓ *skill v1 is included; Task 2 encounters a closed microwave*

Task 2 — put a tomato in microwave (✓ succeeded; selected actions)

```
go to countertop 1 ... go to countertop 4
take tomato 1 from countertop 4
go to microwave 1
open microwave 1                             # destination opened first
move tomato 1 to microwave 1                 # success
```

↓ *the open-if-closed condition is added to skill v2*

Curate — relevant changes in skill v2 after Task 2

Workflow (diff summary). (1)–(4) unchanged. **(5) Check whether the destination receptacle is accessible and open; if it is closed, open it.** ← *new condition* (6) Place the target object on the destination receptacle.

Pitfalls (new sentence). If the destination receptacle is closed, open it before placing the object.

Figure 12: **Case 2: an incremental skill revision.** After a successful task that requires opening the destination, curation adds an "open if closed" condition to the existing placement workflow. The added condition uses generic object and receptacle terms.

Case 2: incremental skill evolution. Both displayed tasks succeed. Task 1 places an object on a destination that requires no `open` action, and skill v1 contains a basic placement workflow. Task 2 requires opening a closed microwave; afterward, the curation output adds an "open the destination if closed" condition to skill v2. This case documents the revision itself, not downstream use of the new condition.