Google

SkillOS: Learning Skill Curation for Self-Evolving Agents

Siru Ouyang[1*], Jun Yan[2†], Yanfei Chen[2], Rujun Han[2], Zifeng Wang[2], Bhavana Dalvi Mishra[2], Rui Meng[2], Chun-Liang Li[2], Yizhu Jiao[1], Kaiwen Zha[3], Maohao Shen[3], Vishy Tirumalashetty[2], George Lee[2], Jiawei Han[1], Tomas Pfister[2] and Chen-Yu Lee[2†]

[1]University of Illinois Urbana-Champaign, [2]Google Cloud AI Research, [3]Massachusetts Institute of Technology

LLM-based agents are increasingly deployed to handle streaming tasks, yet they often remain one-off problem solvers that fail to learn from past interactions. Reusable skills distilled from experience provide a natural substrate for self-evolution, where high-quality skill curation serves as the key bottleneck. Existing approaches either rely on manual skill curation, prescribe heuristic skill operations, or train for short-horizon skill adaptation, but still struggle to learn complex long-term curation policies from indirect and delayed feedback. We propose SkillOS, an experience-driven RL training recipe for learning skill curation in self-evolving agents. SkillOS pairs a frozen *agent executor* that retrieves and applies skills with a trainable *skill curator* that updates an external SkillRepo from accumulated experience. To provide learning signals for curation, we train on grouped task streams based on skill-relevant task dependencies, where earlier trajectories update the SkillRepo, and later related tasks evaluate these updates. We further design composite rewards to better attribute downstream executor feedback to curation decisions. Across multi-turn agentic tasks and single-turn reasoning tasks, SkillOS consistently outperforms memory-free and strong memory-based baselines in both effectiveness and efficiency, with the learned skill curator generalizing across different executor backbones and task domains. Further analyses show that the learned curator produces more targeted skill use, while the evolving SkillRepo develops richer internal structure and higher-level meta-skills over time.

1. Introduction

LLM-based agents (Wang et al., 2024) are increasingly deployed in real-world scenarios, where they must move beyond instantaneous problem-solving toward long-term proficiency (He et al., 2026b). However, the prevailing paradigm of "one-off" task execution limits their utility in streaming settings, where tasks unfold sequentially over time. This makes *self-evolution* (Fang et al., 2025a; Gao et al., 2025) essential: capable agents should not repeatedly start from scratch, but instead continually accumulate, refine, and reuse experience for future tasks.

A key substrate for self-evolution is *procedural memory* (Fang et al., 2025b; Hu et al., 2025; Wu et al., 2025b), specifically, reusable skills (Anthropic, 2025b; Wang et al., 2025c) accumulated from past interactions. In real-world streaming settings (Wu et al., 2024), a skill-based self-evolving agent typically follows a closed-loop workflow: for each new task, it selects relevant skills, uses them to guide execution, and updates its skill collection based on the resulting trajectory. This makes skill curation—the extraction of high-quality lessons and their integration into the skill collection—essential for self-evolving agents.

However, existing skill curation works remain limited. Manually curated skills, such as Anthropic's skills repository (Anthropic, 2025b), demand huge human expertise and cannot scale to the diversity of tasks that agents may encounter. Prompting or heuristic-based methods that dictate memory operations (Qiu et al., 2025; Xu et al., 2025; Zheng et al., 2025) rely on fixed rules and lack downstream performance feedback, preventing them from adapting to the executor's actual needs.

Main contact: siruo2@illinois.edu, {junyann, chenyulee}@google.com
* This work was done while Siru interned at Google Cloud AI Research. † Corresponding authors.

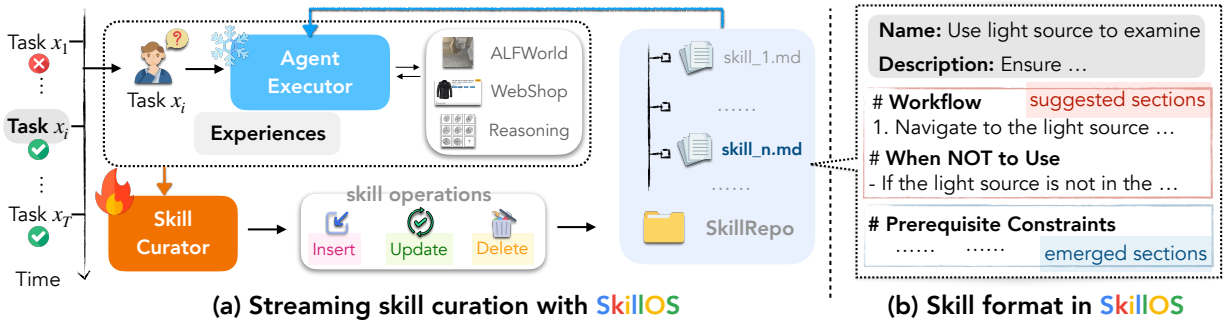


Figure 1 | SkillOS pairs a frozen *Agent Executor* with a trainable *Skill Curator*. The executor retrieves relevant skills from SkillRepo to act; the curator edits the repo (insert/update/delete) based on the resulting experiences, with Markdown as the skill format.

Recent studies explored reinforcement learning (RL) to optimize skill-based agent systems. However, they either focus on teaching agents to *use* skills (Tu et al., 2026; Xia et al., 2026) or optimize skill operations within a short task stream (Wang et al., 2025a; Ye et al., 2026). This limits the density of learning signals available for curating highly reusable skills and mastering complex management operations such as skill update and deletion, which are essential for robust and scalable long-term self-evolution.

To tackle this challenge, we propose SkillOS, an experience-driven RL training recipe to learn the capability of skill curation for self-evolving agents. We study skill curation in a modular multi-agent framework in a streaming setting, where a frozen *agent executor* solves tasks with a skill collection (termed SkillRepo), while a trainable *skill curator* updates and manages this collection through function calls (Figure 1(a)). We represent skills as Markdown files (Anthropic, 2025b) (Figure 1(b)) managed via file I/O operations similar to an operating system (OS). Our recipe features two core designs. *First*, we construct each training instance as a group of related tasks. By mimicking test-time streaming settings, it grounds skill curation in long-term utility: skills induced from earlier experiences are evaluated by their ability to improve later related tasks. *Second*, we design rewards to better attribute environmental feedback to curation decisions, combining task performance with signals for valid function calls, skill quality, and SkillRepo's compactness. Together, these designs turn delayed and indirect supervision into learning signals for skill curation.

We evaluate SkillOS on both multi-turn agentic tasks and single-turn reasoning tasks. Experiments show that SkillOS consistently outperforms memory-free and strong memory-based methods in both effectiveness and efficiency, with up to +9.8% relative performance improvement and −6.0% fewer interaction steps compared to the strongest baseline (Table 1). Our trained skill curator generalizes well across executors and tasks, improving performance even with the Gemini-2.5-Pro executor. Notably, our 8B curator also outperforms Gemini-2.5-Pro when used directly as the curator. Beyond performance gains, our analyses further show that the learned skill curator leads to more targeted and effective skill utilization, while the skills in SkillRepo evolve into more richly structured Markdown files that encode higher-level meta-skills over time. Together, we establish SkillOS as a practical, modular, and experience-driven RL training recipe for building self-evolving agents.

2. Related Work

Memory for Self-Evolving Agents. Learning from past experiences as procedural memory (Hu et al., 2025; Huang et al., 2026; Shen et al., 2026; Wei et al., 2025; Wu et al., 2025b; Zhang et al., 2024) is a central mechanism for developing self-evolving agents (Fang et al., 2025a; Gao et al., 2025). The central challenge is to encode interaction histories into reusable and retrievable representations.

Case-based representations are the most concrete form in this research line: they store experiences in minimally processed formats, allowing past histories to be replayed directly or reused as in-context exemplars, such as raw trajectories (Wu et al., 2025a; Zheng et al., 2024; Zhou et al., 2025) and abstracted query–response pairs (Islam et al., 2024; Zhao et al., 2024). Another line of work abstracts experiences into higher-level knowledge that is editable, auditable, and composable, reducing reliance on long trajectory replay and improving both cross-task generalization and efficiency. Such strategy-based memory typically consists of reusable workflows (Tang et al., 2025; Wang et al., 2025d), distilled insights (Ho et al., 2025; Huang et al., 2025; Ouyang et al., 2026), and recurring patterns (Kim et al., 2025; Yang et al., 2024). Recently, skills (Alzubi et al., 2026; Kuroki et al., 2025; Li et al., 2026a; Liang et al., 2026; Ling et al., 2026; Wang et al., 2025c; Yang et al., 2026; Zhang et al., 2026a) have emerged as a new agent-native form of memory and an orchestrable capability layer, owing to their modularity and ease of customization. Anthropic conceptualizes each skill as a folder containing instructions, scripts, and supporting resources (Anthropic, 2025a), which has become the most widely adopted design in the current community. Our work follows this design philosophy, simplifying the setting for research purposes by representing each skill as a single *Markdown* file.

Learning Memory and Skill Curation with RL. Training LLM-based agent systems with memory capabilities using RL has become a growing research direction. One research line targets training for long-context management with predefined operations such as compaction (Wang et al., 2025b; Yu et al., 2026; Zhou et al., 2026). Another interesting area focuses more on memory utilization and management by learning additional memory tool-calls (Yan et al., 2025; Zhang et al., 2025a,b) or training policies for different stages, such as memory retrieval (Zhang et al., 2026b). More recently, RL has been applied at various stages of agent skill development. Specifically, SkillRL (Xia et al., 2026) and D2Skill (Tu et al., 2026) teach smaller models to use skills curated from powerful LLMs in an iterative manner. ARISE (Li et al., 2026b) trains a shared policy operating both as skill retriever and worker, with heuristics for skill management. Recent studies have begun to train agents for memory or skill curation (Wang et al., 2025a; Ye et al., 2026), but their supervision is mostly restricted to local adaptation within short task streams. This favors immediately useful operations such as skill insertion, while offering limited signal for complex management operations, such as revising outdated skills and deleting harmful ones. SkillOS instead formulates skill curation as a long-horizon, executor-grounded learning problem. We group related tasks into training instances and combine downstream task outcomes with intermediate rewards, turning delayed and indirect feedback into learning signals for skill curation.

3. Methodology

In this section, we first formalize the problem setting and introduce the multi-agent modular design of SkillOS. We then detail the RL training recipe designed specifically for training the skill curator.

3.1. Streaming Skill Curation with Multi-Agent Modular Design

We consider a *streaming* test-time setting (Wu et al., 2024), where an LLM-based agent is deployed to solve a sequence of tasks $\mathcal{D} = \{x_1, x_2, \ldots, x_T\}$ that arrive over time. At each time stamp t, the agent must solve the current task x_t before observing future tasks, producing an execution trajectory $\xi_t = \{o_1, a_1, \ldots, o_n, a_n\}$, where o and a denote observations and actions, respectively. This setting naturally captures the challenge of self-evolving agents, where the system must distill useful experience from the trajectories of past interactions to improve performance on future tasks, and become more capable over time. Figure 1(a) presents an overview of the system.

Skill Repository. We maintain an external skill repository $\mathcal{S}_t$ at time stamp t, which consists of

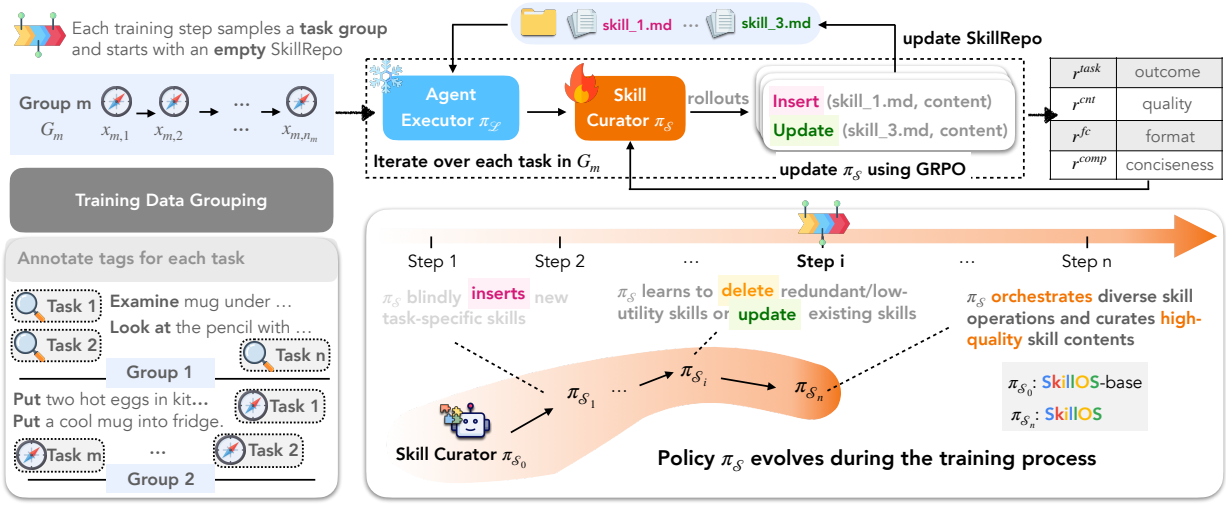


Figure 2 | SkillOS training pipeline. Each training step samples a group of related tasks and initializes an empty SkillRepo. $\pi_{\mathcal{S}}$ is optimized with composite rewards, enabling self-evolution.

N_t reusable skills $\mathcal{S}_t = \{s_t^1, s_t^2, \ldots, s_t^{N_t}\}$. Following the widely adopted `SKILL.md` format (Anthropic, 2025b), each skill is represented as a single Markdown file with two components as shown in Figure 1(b): (i) *YAML frontmatter*, which specifies the skill name and a natural-language description of when the skill should be used, and (ii) *Markdown instructions*, which describe the executable knowledge, workflows, constraints, and reusable heuristics captured by the skill.

Agent Executor. Given a task x_t, a frozen agent executor $\pi_{\mathcal{L}}$ solves the task conditioning on the current environment observation and relevant skills. Specifically, we retrieve a subset of skills $\tilde{\mathcal{S}}_t \subseteq \mathcal{S}_t$ using BM25 (Robertson and Zaragoza, 2009) for each task x_t, and the executor samples actions following $a \sim \pi_{\mathcal{L}}(\cdot \mid x_t, o_t, \tilde{\mathcal{S}}_t)$.

Skill Curator. After the executor completes task x_t, the skill curator $\pi_{\mathcal{S}}$ observes the trajectory ξ_t, the self-judged correctness of the answers/interactions $\mathbb{1}_{\xi_t}$, and a retrieved subset of related skills $\tilde{\mathcal{S}}_t$. It then generates a sequence of structured curation operations $c_t = (u_t^1, \ldots, u_t^{M_t}) \sim \pi_{\mathcal{S}}(\cdot \mid \xi_t, \mathbb{1}_{\xi_t}, \tilde{\mathcal{S}}_t)$, where each operation u_t^m is one of {`insert_skill`, `update_skill`, `delete_skill`}. Each operation is implemented as a function call (detailed signature in Figure 8) that manipulates the skill repository $\mathcal{S}_t$. Applying these operations transforms the repository from $\mathcal{S}_t$ to $\mathcal{S}_{t+1}$ as $\mathcal{S}_{t+1} = \textsc{ApplyOps}(\mathcal{S}_t, c_t)$. The updated repository is then used by the executor on subsequent tasks, forming a closed loop between task execution and experience-driven skill evolution.

3.2. Learning Skill Curation with RL

We optimize the skill curator $\pi_{\mathcal{S}}$ with RL and keep the agent executor $\pi_{\mathcal{L}}$ frozen. The main challenge is indirect and delayed feedback for curation decisions, which is only revealed through $\pi_{\mathcal{L}}$'s performance on future relevant tasks. We address this by constructing grouped training instances (§ 3.2.1) and designing a composite reward (§ 3.2.2) that combines future task outcomes with intermediate signals on operation validity, skill quality, and the conciseness of skills. An overview of the training process is shown in Figure 2.

3.2.1. Training Instance Construction

To provide downstream learning signals for skill curation, we construct each training instance as a group of related tasks that are solved sequentially. Within each group, SkillRepo is updated

by the curator $\pi_\int$ after each task, allowing skills derived from earlier experiences to be evaluated by whether they help solve related future tasks. This also differs from prior work that focuses on short-horizon transfer (Wang et al., 2025a; Ye et al., 2026), where our grouped formulation exposes the curator to longer skill-evolution trajectories and provides denser feedback for learning complex curation operations.

Concretely, for each task x_i in $\mathcal{D} = \{x_i\}_{i=1}^N$, we first annotate each instance with a set of skill-relevant attributes. Formally, for each x_i, we use Gemini-2.5-Pro (Team, 2025a) to produce a set of tags:

$$Z_i = \{z_i^1, z_i^2, \ldots, z_i^{|Z_i|}\},$$

where each attribute z_i captures a salient aspect of the task x_i, such as topic and common pitfalls. For example, in mathematical reasoning, attributes may include labels such as "algebra" or "Fourier transformation". These attributes serve as proxies for task-relatedness and potential skill dependency.

Based on the annotated attributes, we then partition $\mathcal{D}$ into a collection of M task groups using the similarity of attributes of these data samples:

$$\mathcal{D} = \{G_1, G_2, \ldots, G_M\}, \qquad G_m = \{x_{m,1}, x_{m,2}, \ldots, x_{m,|G_m|}\},$$

where all instances within the same group G_m exhibit non-trivial dependency in terms of required skills. Detailed description of data processing and grouping algorithms can be found in Appendix B.2.

3.2.2. Training Loop and Policy Optimization

We employ Grouped Reward Policy Optimization (GRPO Shao et al. (2024)) for its training stability and sample efficiency. The training loop shown in Algorithm 1 optimizes the skill curator policy $\pi_\mathcal{S}$ to maximize a composite reward function over the distribution of generated traces. For a task group $G = (x_1, \ldots, x_{|G|})$, the curator produces a sequence of curation decisions $c = (c_1, \ldots, c_{|G|})$ as the executor proceeds through the group. Each training step, the reward combines four signals:

$$r = \underbrace{r^{\text{task}}}_{\text{task outcome}} + \lambda_{\text{f}} \underbrace{r^{\text{fc}}}_{\text{function call}} + \lambda_{\text{u}} \underbrace{r^{\text{cnt}}}_{\text{content quality}} + \lambda_{\text{c}} \underbrace{r^{\text{comp}}}_{\text{compression}} \tag{1}$$

Task outcome reward. The first task uses an empty SkillRepo, before any curator update occurs. We thus define the task outcome reward as the average success over the remaining tasks as $r^{\text{task}} = \frac{1}{|G|-1} \sum_{i=2}^{|G|} \mathbb{1}(\xi_i)$, which provides executor-grounded signal on downstream performance achieved by the evolving SkillRepo from $\pi_\mathcal{S}$.

Function call reward. The function call reward measures whether the curator produces valid skill operations. For each curation decision c_i, let $\text{Valid}(c_i)$ be the fraction of generated function calls that are valid and successfully executed. We define the function call reward as $r^{\text{fc}} = \frac{1}{|G|} \sum_{i=1}^{|G|} \text{Valid}(c_i)$.

Compression reward. To discourage verbatim trajectory copying, we reward concise repository updates. Let $\mathcal{S}_i$ denote the skill repository after applying c_i, and let χ_i denote the curator input context at position i. We define $r^{\text{comp}} = \frac{1}{|G|} \sum_{i=1}^{|G|} \left(1 - \frac{|\mathcal{S}_i|}{|\chi_i|}\right)$, where $|\mathcal{S}_i|$ and $|\chi_i|$ denote token lengths. This encourages the curator to distill reusable skills rather than store raw trajectories.

Content quality reward. The content quality reward evaluates whether the curated skills are semantically meaningful and likely to be useful for future tasks. Let $\text{Judge}(c_i)$ denote the scalar score assigned by an external judge (Qwen3-32B) c_i, we compute the reward as $r^{\text{cnt}} = \frac{1}{|G|} \sum_{i=1}^{|G|} \text{Judge}(c_i)$.

For each task group G, we sample N independent rollouts of the *entire curation sequence* from $\pi_\mathcal{S}$. Within each rollout, the executor produces trajectory ξ_i using the skill repository $\mathcal{S}^i$ resulting from previous curations $c_{<i}$ till task position i with the same training task group, so different rollouts evolve

Algorithm 1 Training Skill Curator with Task Groups using GRPO

for each training step **do**
  $G = (x_1, \ldots, x_{|G|})$, $\mathcal{S} \leftarrow \emptyset$ ▷ *Sample a task group and initialize SkillRepo*
  for task index $i = 1, \ldots, |G|$ **do**
    $\tilde{\mathcal{S}} \leftarrow \mathrm{BM25}(x_i,\ \mathcal{S})$ ▷ *Retrieve relevant skills*
    $\xi_i \leftarrow \textsc{RunTask}\left(\tilde{\mathcal{S}},\ \pi_{\mathcal{L}},\ x_i\right)$ ▷ *Run inference on frozen executor*
    $c_i \sim \pi_{\mathcal{S}}\left(\cdot \,\middle|\, \xi_i, \tilde{\mathcal{S}}\right)$ ▷ *Sample a rollout from skill curator*
    $\mathcal{S} \leftarrow \textsc{ApplyOps}(\mathcal{S},\ c_i)$ ▷ *Apply* `insert/update/delete`
  end for
  $r \leftarrow \textsc{CalculateReward}(\xi, c)$
  Update $\pi_{\mathcal{S}}$ ▷ *Update skill curator using GRPO*
end for

different repository histories. The GRPO advantage is computed as: $A^n = r^n - \frac{1}{N}\sum_{n'=1}^{N} r^{n'}$, where r^n is the composite reward (Eq. 1) for the n-th rollout. We optimize $\pi_{\mathcal{S}}$ with a clipped surrogate objective over all curation steps $i = 1, \ldots, |G|$:

$$\mathcal{L} = \mathbb{E}_n[\min(\rho^n A^n,\ \mathrm{clip}(\rho^n,\ 1-\epsilon,\ 1+\epsilon)A^n)] \tag{2}$$

where $\rho^n = \pi_{\mathcal{S}}(c^n \mid \chi) \,/\, \pi_{\theta_{old}}(c^n \mid \chi)$ is the importance ratio. The advantage A^n is assigned uniformly to all tokens in c^n, and we discard the KL term in GRPO to encourage policy exploration.

4. Experiments

We conduct experiments on both multi-turn agentic tasks and single-turn reasoning tasks, in line with prior work (Wei et al., 2025; Xia et al., 2026; Ye et al., 2026). We additionally show that the trained skill curator transfers across agent executors and task domains, highlighting its flexibility and generalizability.

4.1. Setup

We briefly discuss the experiment setup throughout this paper. Full description of datasets, implementations, baselines, and evaluations can be found in Appendix B.

Dataset. For agentic tasks, we conduct experiments on ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2022). ALFWorld is a text-based interactive environment aligned with the ALFRED embodied AI benchmark, where agents must complete household tasks through textual navigation and object manipulation. WebShop simulates an online shopping environment in which agents navigate a realistic web interface to identify and purchase products that satisfy user-specified requirements. For each benchmark, we train SkillOS on its training split where Z_i is the default task type annotations, and evaluate on the corresponding test set. In addition to agentic tasks, we also benchmark for single-turn reasoning tasks, including AIME24, AIME25, and GPQA-Diamond (Rein et al., 2024). Training data are constructed from DeepMath-103k (He et al., 2026a), where we randomly sample a subset of 33,000 data points.

Evaluation Configurations. We evaluate all methods across two dimensions, *effectiveness* and *efficiency*. For effectiveness, we measure the success rate (SR) and accuracy for agentic tasks and reasoning tasks, respectively. For efficiency, we compute the number of execution steps per agentic task and the number of tokens per reasoning problem, respectively. We compare SkillOS with three categories of baselines: (i) a memory-free agent (No Memory); (ii) existing memory-based methods, including

Table 1 | Experiment results on ALFWorld benchmark. Success rate (SR ↑) and the number of steps (Steps ↓) are reported on 6 subsets with 3 different frozen executors.

Methods	Curator π_S	Pick (35)	Look (13)	Clean (27)	Heat (16)	Cool (25)	Pick2 (24)	Avg. SR (140)	Steps
Executor $\pi_\mathcal{L}$: Qwen3-8B									
No Memory	None	78.1 $_{1.6}$	46.2 $_{7.7}$	33.3 $_{13.4}$	37.5 $_{10.8}$	29.3 $_{6.1}$	47.2 $_{6.4}$	47.9 $_{1.2}$	21.1
ReasoningBank	❄ Qwen3-8B	83.8 $_{0.0}$	48.7 $_{7.2}$	49.4 $_{16.2}$	39.6 $_{4.4}$	41.3 $_{8.5}$	54.2 $_{8.8}$	55.7 $_{3.1}$	20.1
MemP	❄ Qwen3-8B	80.0 $_{5.7}$	43.6 $_{4.4}$	24.7 $_{4.3}$	33.3 $_{3.6}$	38.7 $_{6.1}$	48.6 $_{6.4}$	49.7 $_{0.7}$	21.0
SkillOS-base	❄ Qwen3-8B	79.0 $_{8.7}$	41.0 $_{4.4}$	45.7 $_{4.3}$	37.5 $_{9.5}$	38.7 $_{4.0}$	55.6 $_{2.1}$	53.1 $_{2.5}$	20.4
SkillOS-gemini	❄ Gemini-2.5-Pro	77.1 $_{6.0}$	53.8 $_{6.1}$	37.0 $_{6.4}$	37.5 $_{9.5}$	36.0 $_{3.2}$	50.0 $_{6.7}$	50.7 $_{3.6}$	20.8
SkillOS	🔥 Qwen3-8B	**85.7** $_{3.3}$	**56.4** $_{7.7}$	**54.3** $_{8.6}$	**43.8** $_{9.5}$	**46.7** $_{2.3}$	**62.5** $_{6.4}$	**61.2** $_{4.6}$	**18.9**
Executor $\pi_\mathcal{L}$: Qwen3-32B									
No Memory	None	80.0 $_{2.9}$	69.2 $_{0.0}$	45.6 $_{7.7}$	37.5 $_{16.5}$	42.7 $_{6.1}$	43.1 $_{2.4}$	54.5 $_{2.5}$	20.3
ReasoningBank	❄ Qwen3-8B	86.7 $_{3.0}$	71.8 $_{5.4}$	50.6 $_{6.3}$	45.8 $_{13.3}$	52.0 $_{8.9}$	51.4 $_{5.1}$	61.4 $_{2.5}$	18.7
MemP	❄ Qwen3-8B	80.0 $_{2.9}$	76.9 $_{0.0}$	44.4 $_{7.4}$	37.5 $_{10.8}$	42.7 $_{2.3}$	47.2 $_{6.4}$	55.7 $_{3.7}$	20.0
SkillOS-base	❄ Qwen3-8B	82.9 $_{2.9}$	69.2 $_{11.8}$	48.1 $_{2.1}$	50.0 $_{9.7}$	48.0 $_{14.4}$	52.8 $_{11.0}$	59.8 $_{3.0}$	19.2
SkillOS-gemini	❄ Gemini-2.5-Pro	**97.1** $_{3.0}$	76.9 $_{5.4}$	55.6 $_{6.0}$	43.8 $_{11.3}$	40.0 $_{5.7}$	54.2 $_{4.9}$	63.6 $_{4.2}$	18.1
SkillOS	🔥 Qwen3-8B	91.4 $_{3.3}$	**76.9** $_{4.4}$	**59.3** $_{8.6}$	**56.3** $_{12.5}$	**57.3** $_{10.1}$	**62.5** $_{4.2}$	**68.6** $_{5.7}$	**17.3**
Executor $\pi_\mathcal{L}$: Gemini-2.5-pro									
No Memory	None	90.5 $_{3.2}$	66.7 $_{5.1}$	48.1 $_{10.2}$	39.6 $_{17.1}$	68 $_{7.4}$	68.1 $_{3.8}$	66.4 $_{2.0}$	17.7
ReasoningBank	❄ Qwen3-8B	91.4 $_{3.4}$	61.5 $_{4.1}$	63.0 $_{9.3}$	39.6 $_{10.3}$	70.7 $_{3.2}$	76.4 $_{8.5}$	71.4 $_{2.9}$	16.0
MemP	❄ Qwen3-8B	95.2 $_{2.1}$	**74.4** $_{6.8}$	61.7 $_{7.6}$	56.3 $_{12.4}$	76.0 $_{6.2}$	68.1 $_{8.5}$	74.3 $_{3.4}$	15.2
SkillOS-base	❄ Qwen3-8B	91.4 $_{1.6}$	69.2 $_{7.7}$	56.8 $_{5.7}$	54.2 $_{13.7}$	72.0 $_{4.0}$	66.7 $_{11.0}$	70.7 $_{3.0}$	16.3
SkillOS-gemini	❄ Gemini-2.5-Pro	94.3 $_{5.7}$	69.2 $_{0.0}$	**77.8** $_{5.7}$	**75.0** $_{16.5}$	80.0 $_{12.2}$	66.7 $_{2.4}$	79.3 $_{2.6}$	14.9
SkillOS	🔥 Qwen3-8B	**95.2** $_{2.9}$	71.8 $_{7.7}$	74.1 $_{13.0}$	72.9 $_{10.1}$	**77.3** $_{6.1}$	**77.8** $_{10.0}$	**80.2** $_{3.1}$	**14.8**

ReasoningBank (Ouyang et al., 2026), which distills reusable insights from past experiences, and MemP (Fang et al., 2025b), which induces procedural memory with advanced memory-management strategies; and (iii) internal variants of our framework, including SkillOS-base, which uses the initial skill curator without RL training, and SkillOS-gemini, which uses Gemini-2.5-Pro to directly perform skill curation instead of learning the curator with RL. All prompts used can be found in Appendix A.

Implementation Details. We use Qwen3-8B (Team, 2025b) as the base model for π_S. The frozen executor is also instantiated with Qwen3-8B during training. We train our model using GRPO with a learning rate 1×10^{-6}, batch size 32, and group size 8. Training is conducted on 16 H100 GPUs using the `verl` framework (Sheng et al., 2024). The full training process takes approximately 3 days for ALFWorld, 2.5 days for reasoning tasks, and 5 days for WebShop. For testing, we additionally include Qwen3-32B, Gemini-2.5-Pro (Team, 2025a), and Gemini-3.1-Flash-Lite (Appendix C.1) as executors to evaluate the generalization of SkillOS under different executor scales and architectures. Task outcome signal $\mathbb{1}_{\xi_t}$ is obtained via LLM-as-a-judge with the frozen agent executor (prompt shown in Appendix A). We use ReAct (Yao et al., 2023) for agent execution and CoT (Wei et al., 2022a) for reasoning tasks. For the reward function, we set $\lambda_f = 1.0$, $\lambda_u = 0.1$, and $\lambda_c = 0.05$. We report averaged performance and standard deviation over 3 runs.

4.2. Main Results

Tables 1 and 2 summarize the results for different benchmarks with Qwen3-8B as the skill curator on various agent executors. Based on the results, we have the following observations.

SkillOS achieves strong performance gains across benchmarks. Across all three benchmarks,

Table 2 | Experiment results on WebShop and single-turn reasoning tasks for 3 different frozen executors. For WebShop, the averaged score, success rate (SR ↑), and the number of steps (Steps ↓) are reported. For reasoning tasks, accuracy (Acc. ↑) is reported on three datasets.

Methods	Curator π_S	WebShop			Reasoning			
		Score	SR	Steps	AIME24	AIME25	GPQA	Avg. Acc
Executor $\pi_\mathcal{L}$: Qwen3-8B								
No Memory	None	33.3 $_{0.7}$	9.8 $_{0.5}$	20.3	76.0 $_{6.9}$	71.1 $_{10.7}$	61.8 $_{1.1}$	69.6 $_{4.7}$
ReasoningBank	❄ Qwen3-8B	35.4 $_{1.1}$	11.4 $_{0.9}$	20.5	75.4 $_{5.0}$	73.2 $_{10.8}$	60.3 $_{3.9}$	69.6 $_{2.5}$
MemP	❄ Qwen3-8B	35.7 $_{0.9}$	12.0 $_{0.5}$	21.3	75.6 $_{5.1}$	71.1 $_{5.1}$	60.6 $_{4.0}$	69.1 $_{4.0}$
SkillOS-base	❄ Qwen3-8B	38.6 $_{0.9}$	13.6 $_{0.8}$	20.1	75.6 $_{5.1}$	71.9 $_{6.9}$	59.3 $_{2.5}$	68.9 $_{2.6}$
SkillOS-gemini	❄ Gemini-2.5-pro	38.1 $_{1.0}$	13.2 $_{0.9}$	19.6	73.3 $_{1.3}$	71.3 $_{1.9}$	57.6 $_{2.8}$	67.4 $_{0.8}$
SkillOS	🔥 Qwen3-8B	**40.6** $_{0.7}$	**16.5** $_{0.7}$	**19.4**	**80.0** $_{3.3}$	**76.7** $_{5.8}$	**64.6** $_{1.3}$	**73.8** $_{1.8}$
Executor $\pi_\mathcal{L}$: Qwen3-32B								
No Memory	None	41.5 $_{0.5}$	12.2 $_{0.3}$	17.0	81.4 $_{1.3}$	72.2 $_{3.8}$	68.4 $_{2.0}$	74.0 $_{1.9}$
ReasoningBank	❄ Qwen3-32B	40.4 $_{0.8}$	11.2 $_{1.1}$	17.9	81.1 $_{9.6}$	75.6 $_{5.9}$	66.9 $_{1.2}$	74.9 $_{2.2}$
MemP	❄ Qwen3-32B	30.7 $_{0.7}$	10.1 $_{0.6}$	17.4	82.2 $_{5.1}$	76.7 $_{0.0}$	66.5 $_{2.3}$	75.1 $_{2.1}$
SkillOS-base	❄ Qwen3-8B	43.4 $_{0.8}$	12.3 $_{1.0}$	16.8	80.0 $_{3.3}$	75.6 $_{10.2}$	67.7 $_{1.5}$	74.7 $_{3.3}$
SkillOS-gemini	❄ Gemini-2.5-pro	45.2 $_{1.0}$	13.2 $_{1.1}$	16.6	77.8 $_{6.7}$	74.4 $_{1.9}$	66.2 $_{0.6}$	73.2 $_{2.6}$
SkillOS	🔥 Qwen3-8B	**49.2** $_{1.2}$	**16.5** $_{0.6}$	**15.9**	**85.6** $_{1.9}$	**81.1** $_{3.3}$	**72.4** $_{3.0}$	**79.7** $_{1.6}$
Executor $\pi_\mathcal{L}$: Gemini-2.5-pro								
No Memory	None	48.6 $_{0.3}$	38.4 $_{0.5}$	19.5	85.6 $_{1.9}$	80.0 $_{6.7}$	79.9 $_{1.5}$	81.8 $_{2.8}$
ReasoningBank	❄ Gemini-2.5-pro	50.8 $_{1.5}$	40.2 $_{1.3}$	19.2	85.6 $_{5.1}$	84.4 $_{6.7}$	80.4 $_{2.1}$	83.5 $_{2.1}$
MemP	❄ Gemini-2.5-pro	51.3 $_{1.2}$	39.8 $_{1.0}$	19.4	83.3 $_{6.9}$	76.7 $_{5.8}$	81.8 $_{3.4}$	80.6 $_{3.2}$
SkillOS-base	❄ Qwen3-8B	52.8 $_{1.0}$	39.6 $_{0.8}$	19.0	87.8 $_{3.3}$	83.3 $_{1.9}$	82.8 $_{2.7}$	84.6 $_{1.8}$
SkillOS-gemini	❄ Gemini-2.5-pro	54.7 $_{1.0}$	41.0 $_{1.2}$	**17.8**	90.0 $_{5.1}$	85.6 $_{7.7}$	80.7 $_{5.5}$	85.4 $_{3.5}$
SkillOS	🔥 Qwen3-8B	**56.0** $_{0.7}$	**41.3** $_{0.8}$	18.3	**92.2** $_{2.4}$	**86.7** $_{3.5}$	**86.8** $_{2.1}$	**88.6** $_{1.5}$

SkillOS consistently outperforms both memory-free and memory-based baselines, showing that the gains come from *learning to manage and evolve* skills rather than from maintaining a static collection. On ALFWorld, SkillOS improves the average success rate from 55.7 to 61.2 over the strongest baseline ReasoningBank with Qwen3-8B as the executor; similar trends hold on WebShop and reasoning tasks. Strikingly, our RL-trained 8B curator even surpasses SkillOS-gemini, despite the latter using a far stronger frontier model as the curator, demonstrating that targeted training of a small curator can outweigh raw model scale. The benefits brought by RL training are also compounded with executor capacity, yielding +9.5 absolute improvement with Gemini-2.5-Pro versus +7.9 with Qwen3-8B for ALFworld, compared with SkillOS-base.

SkillOS is more efficient, requiring fewer interactions and lower execution cost. The gains of SkillOS are accompanied by better efficiency rather than longer trajectories. On ALFWorld, SkillOS reduces the average interaction steps by 2.2, 3.0, and 3.1 compared with "no memory" setting with 3 executors, consistently outperforming all memory-based baselines. This trend extends to WebShop, where SkillOS secures higher success rates with fewer environment interactions. These results indicate that the learned skill manager enables the executor to identify procedural shortcuts and bypass redundant exploration. Rather than relying on additional trial-and-error, SkillOS improves performance by distilling experience into direct, actionable expertise that simplifies task execution.

The gains differ between agentic and reasoning tasks, reflecting different forms of reusable skills. A notable trend is that the gains of SkillOS are generally larger on multi-turn agentic benchmarks than on single-turn reasoning tasks. We hypothesize that this difference arises from how reusable skills manifest across task types. Agentic tasks naturally expose procedural regularities, such

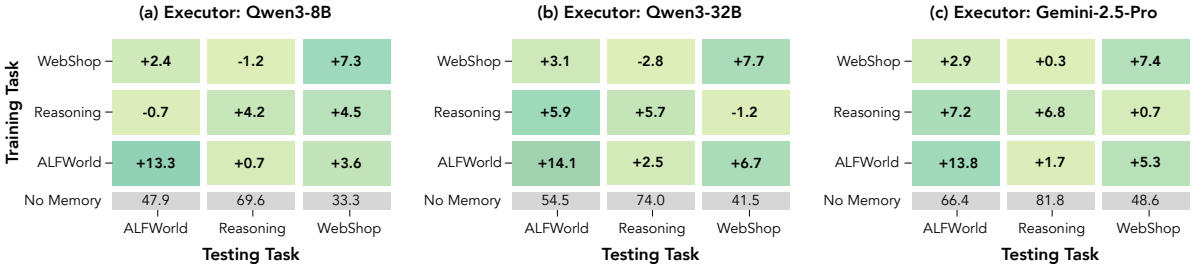


Figure 3 | Cross-task generalization results of SkillOS with (a) Qwen3-8B, (b) Qwen3-32B, and (c) Gemini-2.5-Pro as frozen executors. We plot relative improvement with baselines from least to most.

as action ordering, exploration strategies, recovery behaviors, and environment-specific constraints, which can be repeatedly composed and refined across task streams. Reasoning tasks also benefit from skill curation, but their reusable knowledge often appears at a more abstract level, such as decomposition heuristics, constraint formulation, or verification patterns, rather than as directly reusable action procedures. As a result, SkillOS still improves reasoning performance, while the gains are typically smaller than those observed on agentic benchmarks. We provide a case study demonstrating skills curated for different tasks in Figure 17.

4.3. Generalization of SkillOS

SkillOS is transferable and remains effective for different agent executors. During training, we use Qwen3-8B as the executor. To test whether SkillOS brings improvement for executors that are not seen in training, we pair the trained skill curator with different executors. As shown in Table 1 and 2, SkillOS consistently improves a wide range of frozen executors across benchmarks, from open-source models (Qwen3-8B, Qwen3-32B) to frontier models (Gemini-2.5-Pro). On ALFWorld, it lifts the average success rate of Qwen3-8B from 47.9 to 61.2 and Gemini-2.5-Pro from 66.4 to 80.2, demonstrating compatibility with executors of varying capacity. Notably, using Gemini-2.5-Pro directly as the curator (SkillOS-gemini) underperforms our trained curator, especially when paired with the smaller Qwen3-8B executor. This highlights a curator-executor mismatch: stronger reasoning ability alone does not guarantee effective skill curation, as frontier-generated skills may be misaligned with the executor's capacity or usage patterns. By contrast, SkillOS learns executor-grounded curation behaviors through RL, producing skills that better match the downstream agent.

SkillOS delivers consistent performance improvement when generalized to different task domains. Figure 3 shows that the skill curator learned by SkillOS transfers well across different tasks. While training and testing on the same task often gives the strongest gain, most off-diagonal entries still bring performance improvement over baselines, indicating that SkillOS captures reusable skills beyond task-specific heuristics. Specifically, skill curator π_s learned from reasoning tasks transfer particularly well to the two agentic tasks, likely because they contain more abstract and high-level strategies, such as decomposition, verification, and adaptive planning, which are broadly useful across settings. In contrast, skills learned from WebShop or ALFWorld are more tied to environment-specific knowledge, making them less transferable across tasks.

5. Analysis

Beyond performance, we analyze *why* SkillOS works, focusing on design choices, evolution of curator's behaviors and contents in SkillRepo, and the role of curated skills in task success. Additional analyses are included in Appendix C.

Ablation Studies. We ablate two components of SkillOS: (i) auxiliary rewards in Eq. 1, and (ii) grouped task streams in § 3.2.1. Experiments are conducted on ALFWorld, with Qwen3-8B used as both the curator and executor. As shown in Table 3, removing either reward component hurts performance. Without the content-quality reward, the success rate drops from 61.2 to 58.6, showing the importance of intermediate supervision for guiding skill updates in a pipelined system. Removing the compression reward causes a smaller but consistent drop, suggesting that concise repositories are easier for the executor to use. The most significant degradation comes from using random task sequences (w/o grouping), which lowers the success rate to 57.3. This highlights the importance of training on grouped task streams, in which curation decisions are learned from their downstream impact on related future tasks.

Table 3 | Ablation results of reward design on the ALFWorld dataset.

Methods	Avg. SR	Steps
SkillOS-GRPO	61.2	18.9
w/o r^{cnt}	58.6	20.1
w/o r^{comp}	60.0	19.3
w/o grouping	57.3	20.6

Behaviors of Skill Curator. To better understand how the behavior of the skill curator evolves during training, we analyze the distribution of its three skill operations from rollouts at different training steps: `insert_skill`, `update_skill`, and `delete_skill`. Figure 4 plots the proportion of each operation. At the beginning of training, `insert` overwhelmingly dominates, indicating that the model is primarily focused on populating the skill repository with new knowledge distilled from experience. As training progresses, however, `update` becomes increasingly frequent, while `insert` steadily declines. This suggests that the skill curator gradually moves from plain expansion of skills to refining existing skills. Meanwhile, `delete` remains a relatively small fraction throughout training with a slightly growing trend, showing the effectiveness of rewarding conciseness of SkillRepo. Instead, the dominant form of adaptation is to revise and consolidate previously acquired skills.

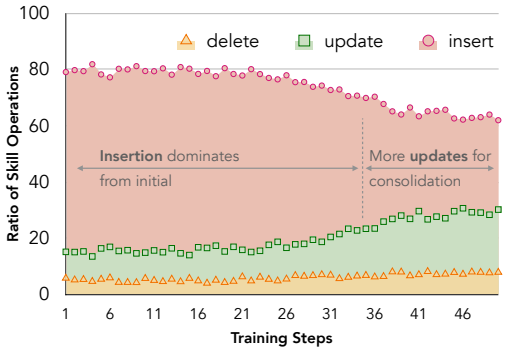
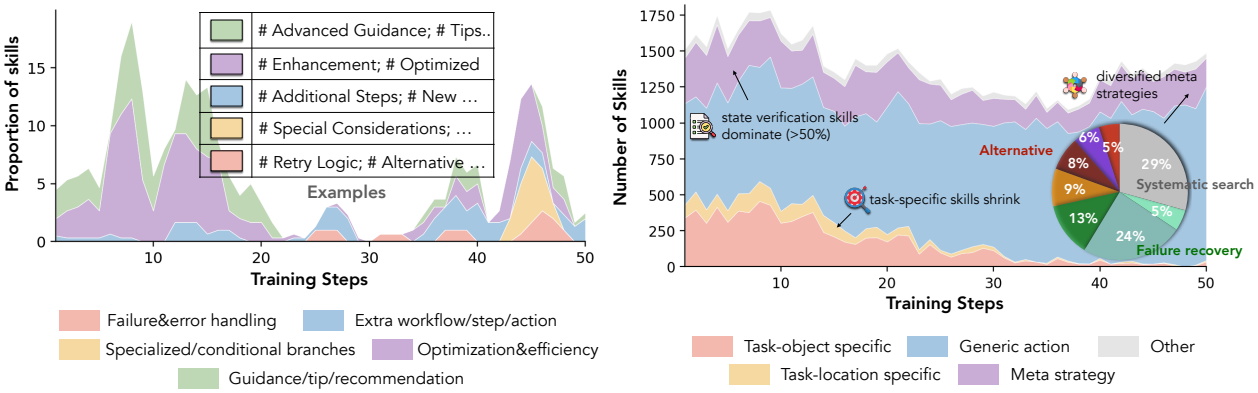


Figure 4 | Behaviors of the skill curator w.r.t. skill operations during training.



(a) Emergence of new markdown sections

(b) Emergence of meta-strategy skills

Figure 5 | Evolution dynamics of the curated skills under RL training.

Skill Evolution Dynamics. Beyond aggregate performance, we examine how the skill repository evolves during RL training. We focus on two emergent phenomena: (i) new Markdown sections within individual skills, and (ii) higher-level meta-skills that capture reusable principles across tasks. Figure 5(a) shows that early in training, the curator tends to introduce generic sections such as additional guidance, tips, or recommendations, which often make skills more verbose without

substantially improving their operational value. As training progresses, these additions shift toward more actionable structures, such as failure-handling logic and conditional branches that specify when to deviate from the default workflow. This suggests that RL gradually steers the curator from superficial enrichment toward execution-oriented skill refinement. Figure 5(b) further shows that evolution occurs not only within individual skills, but also in the global organization of the repository. Early repositories are dominated by narrow, task-specific skills, whereas later repositories contain a more diverse set of meta-strategy skills covering verification, fallback planning, system search, and strategy adjustment. This indicates that the learned curator does not merely accumulate skills, but progressively expands the repository's strategic space, shifting it from isolated task-local procedures toward more compositional cross-task control knowledge.

Attribution of Skill Usage. To better understand whether the gains of SkillOS come from the evolved skills, we analyze how skills are used during evaluation. We consider 4 complementary metrics: (i) *skill usage rate*, the fraction of examples where the agent invokes at least one skill; (ii) *successful skill usage rate*, the success rate among examples that use skills; (iii) *skill coverage*, the fraction of the skill collection that are actually used; and (iv) the *average number of skills used per example*, which measures the degree of skill reliance. Figure 6 reports results on ALFWorld. Compared with the baseline, SkillOS invokes skills on *all* evaluation examples and achieves a higher success rate, indicating that the evolved skills contribute directly to task solving. Also, a larger fraction of the skill curated by SkillOS is used, showing that RL training improves the overall utility of the curated SkillRepo. Meanwhile, SkillOS uses fewer skills per example, suggesting that gains come from more precise skill selection rather than more skill context.

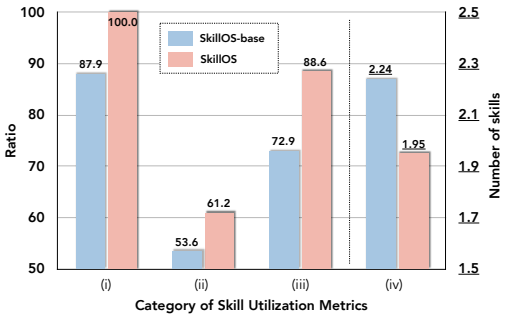


Figure 6 | Comparison of skill utilization statistics on ALFWorld.

6. Conclusion

We presented SkillOS, an RL training recipe for learning skill curation in self-evolving agents. By decoupling the *skill curator* from the *agent executor*, SkillOS enables modular skill curation without retraining the underlying executor. Through grouped task streams and executor-grounded rewards, SkillOS optimizes curation decisions by their downstream impact on future tasks. Across diverse benchmarks and LLM backbones, SkillOS consistently improves both performance and efficiency. Further analyses show that trained skill curation can outperform frontier models' zero-shot curation ability and generalize across settings, highlighting modular, trained skill curation as a practical path toward agents that self-evolve from experience.

7. Acknowledgments

We thank Zilin Xiao, I-Hung Hsu, Zexue He, and members from Google Cloud AI Research for their valuable feedback during the preparation of the paper. Siru was supported by the Molecule Maker Lab Institute: An AI Research Institutes program supported by NSF under Award No. 2019897.

References

S. Alzubi, N. Provenzano, J. Bingham, W. Chen, and T. Vu. Evoskill: Automated skill discovery for multi-agent systems, 2026. URL https://arxiv.org/abs/2603.02766.

Anthropic. Agent skills, 2025a. URL https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview. Claude API Docs. Accessed: 2026-04-01.

Anthropic. Skills. https://github.com/anthropics/skills, 2025b.

J. Fang, Y. Peng, X. Zhang, Y. Wang, X. Yi, G. Zhang, Y. Xu, B. Wu, S. Liu, Z. Li, et al. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *ArXiv preprint*, abs/2508.07407, 2025a. URL https://arxiv.org/abs/2508.07407.

R. Fang, Y. Liang, X. Wang, J. Wu, S. Qiao, P. Xie, F. Huang, H. Chen, and N. Zhang. Memp: Exploring agent procedural memory. *ArXiv preprint*, abs/2508.06433, 2025b. URL https://arxiv.org/abs/2508.06433.

L. Feng, Z. Xue, T. Liu, and B. An. Group-in-group policy optimization for llm agent training. *ArXiv preprint*, abs/2505.10978, 2025. URL https://arxiv.org/abs/2505.10978.

H.-a. Gao, J. Geng, W. Hua, M. Hu, X. Juan, H. Liu, S. Liu, J. Qiu, X. Qi, Y. 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*, abs/2507.21046, 2025. URL https://arxiv.org/abs/2507.21046.

D. Guo, D. Yang, H. Zhang, J. Song, P. Wang, Q. Zhu, R. Xu, R. Zhang, S. Ma, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *ArXiv preprint*, abs/2501.12948, 2025. URL https://arxiv.org/abs/2501.12948.

Z. He, T. Liang, J. Xu, Q. Liu, X. Chen, Y. Wang, L. Song, D. Yu, Z. Liang, W. Wang, Z. Zhang, R. Wang, Z. Tu, H. Mi, and D. Yu. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=kHB5Te5IWm.

Z. He, Y. Wang, C. Zhi, Y. Hu, T.-P. Chen, L. Yin, Z. Chen, T. A. Wu, S. Ouyang, Z. Wang, et al. Memoryarena: Benchmarking agent memory in interdependent multi-session agentic tasks. *ArXiv preprint*, abs/2602.16313, 2026b. URL https://arxiv.org/abs/2602.16313.

M. Ho, C. Si, Z. Feng, F. Yu, Y. Yang, Z. Liu, Z. Hu, and L. Qin. Arcmemo: Abstract reasoning composition with lifelong LLM memory. *ArXiv preprint*, abs/2509.04439, 2025. URL https://arxiv.org/abs/2509.04439.

Y. Hu, S. Liu, Y. Yue, G. Zhang, B. Liu, F. Zhu, J. Lin, H. Guo, S. Dou, Z. Xi, et al. Memory in the age of ai agents. *ArXiv preprint*, abs/2512.13564, 2025. URL https://arxiv.org/abs/2512.13564.

T. Huang, K. Basu, I. Abdelaziz, P. Kapanipathi, J. May, and M. Chen. R2D2: Remembering, replaying and dynamic decision making with a reflective agentic memory. In W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30318–30330, Vienna, Austria, 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1464. URL https://aclanthology.org/2025.acl-long.1464/.

W.-C. Huang, W. Zhang, Y. Liang, Y. Bei, Y. Chen, T. Feng, X. Pan, Z. Tan, Y. Wang, T. Wei, et al. Rethinking memory mechanisms of foundation agents in the second half. *ArXiv preprint*, abs/2602.06052, 2026. URL https://arxiv.org/abs/2602.06052.

M. A. Islam, M. E. Ali, and M. R. Parvez. MapCoder: Multi-agent code generation for competitive problem solving. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4912–4944,

Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.269. URL https://aclanthology.org/2024.acl-long.269/.

N. Kim, K. T.-i. Ong, Y. Hwang, M. Kang, I. Jihn, G. Kim, M. Kim, and J. Yeo. PRINCIPLES: Synthetic strategy memory for proactive dialogue agents. In C. Christodoulopoulos, T. Chakraborty, C. Rose, and V. Peng, editors, *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 21329–21368, Suzhou, China, 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.1164. URL https://aclanthology.org/2025.findings-emnlp.1164/.

S. Kuroki, T. Nakamura, T. Akiba, and Y. Tang. Agent skill acquisition for large language models via cycleQD. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Kvdh12wGC0.

X. Li, W. Chen, Y. Liu, S. Zheng, X. Chen, Y. He, Y. Li, B. You, H. Shen, J. Sun, S. Wang, Q. Zeng, D. Wang, X. Zhao, Y. Wang, R. B. Chaim, Z. Di, Y. Gao, J. He, Y. He, L. Jing, L. Kong, X. Lan, J. Li, S. Li, Y. Li, Y. Lin, X. Liu, X. Liu, H. Lyu, Z. Ma, B. Wang, R. Wang, T. Wang, W. Ye, Y. Zhang, H. Xing, Y. Xue, S. Dillmann, and H. Lee. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *ArXiv preprint*, abs/2602.12670, 2026a. URL https://arxiv.org/abs/2602.12670.

Y. Li, R. Miao, Z. Qi, and T. Lan. Arise: Agent reasoning with intrinsic skill evolution in hierarchical reinforcement learning. *ArXiv preprint*, abs/2603.16060, 2026b. URL https://arxiv.org/abs/2603.16060.

Y. Liang, R. Zhong, H. Xu, C. Jiang, Y. Zhong, R. Fang, J.-C. Gu, S. Deng, Y. Yao, M. Wang, et al. Skillnet: Create, evaluate, and connect ai skills. *ArXiv preprint*, abs/2603.04448, 2026. URL https://arxiv.org/abs/2603.04448.

G. Ling, S. Zhong, and R. Huang. Agent skills: A data-driven analysis of claude skills for extending large language model functionality. *ArXiv preprint*, abs/2602.08004, 2026. URL https://arxiv.org/abs/2602.08004.

S. Ouyang, J. Yan, I.-H. Hsu, Y. Chen, K. Jiang, Z. Wang, R. Han, L. Le, S. Daruki, X. Tang, V. Tirumalashetty, G. Lee, M. Rofouei, H. Lin, J. Han, C.-Y. Lee, and T. Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jL7fwchScm.

J. Qiu, X. Qi, T. Zhang, X. Juan, J. Guo, Y. Lu, Y. Wang, Z. Yao, Q. Ren, X. Jiang, et al. Alita: Generalist agent enabling scalable agentic reasoning with minimal predefinition and maximal self-evolution. *ArXiv preprint*, abs/2505.20286, 2025. URL https://arxiv.org/abs/2505.20286.

N. Reimers and I. Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410.

D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

S. Robertson and H. Zaragoza. *The probabilistic relevance framework: BM25 and beyond*, volume 4. Now Publishers Inc, 2009.

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, M. Zhang, Y. K. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv preprint*, abs/2402.03300, 2024. URL https://arxiv.org/abs/2402.03300.

M. Shen, K. Zha, Z. He, Z.-W. Hong, S. Ouyang, J. J. Ryu, P. Sattigeri, S. Diggavi, and G. Wornell. Decocted experience improves test-time inference in llm agents. *ArXiv preprint*, abs/2604.04373, 2026. URL https://arxiv.org/abs/2604.04373.

G. Sheng, C. Zhang, Z. Ye, X. Wu, W. Zhang, R. Zhang, Y. Peng, H. Lin, and C. Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

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

X. Tang, T. Qin, T. Peng, Z. Zhou, D. Shao, T. Du, X. Wei, P. Xia, F. Wu, H. Zhu, G. Zhang, J. Liu, X. Wang, S. Hong, C. Wu, H. Cheng, C. Wang, and W. Zhou. Agent KB: leveraging cross-domain experience for agentic problem solving. *ArXiv preprint*, abs/2507.06229, 2025. URL https://arxiv.org/abs/2507.06229.

G. Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *ArXiv preprint*, abs/2507.06261, 2025a. URL https://arxiv.org/abs/2507.06261.

Q. Team. Qwen3 technical report. *ArXiv preprint*, abs/2505.09388, 2025b. URL https://arxiv.org/abs/2505.09388.

S. Tu, C. Xu, Q. Zhang, Y. Zhang, X. Lan, L. Li, and D. Zhao. Dynamic dual-granularity skill bank for agentic rl. *ArXiv preprint*, abs/2603.28716, 2026. URL https://arxiv.org/abs/2603.28716.

J. Wang, Q. Yan, Y. Wang, Y. Tian, S. S. Mishra, Z. Xu, M. Gandhi, P. Xu, and L. L. Cheong. Reinforcement learning for self-improving agent with skill library. *ArXiv preprint*, abs/2512.17102, 2025a. URL https://arxiv.org/abs/2512.17102.

L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, W. X. Zhao, Z. Wei, and J. Wen. A survey on large language model based autonomous agents. *Frontiers Comput. Sci.*, 18(6):186345, 2024. doi: 10.1007/S11704-024-40231-1. URL https://doi.org/10.1007/s11704-024-40231-1.

Y. Wang, R. Takanobu, Z. Liang, Y. Mao, Y. Hu, J. McAuley, and X. Wu. Mem-{\alpha}: Learning memory construction via reinforcement learning. *ArXiv preprint*, abs/2509.25911, 2025b. URL https://arxiv.org/abs/2509.25911.

Z. Z. Wang, A. Gandhi, G. Neubig, and D. Fried. Inducing programmatic skills for agentic tasks. In *Second Conference on Language Modeling*, 2025c. URL https://openreview.net/forum?id=lsAY6fWsog.

Z. Z. Wang, J. Mao, D. Fried, and G. Neubig. Agent workflow memory. In *Forty-second International Conference on Machine Learning*, 2025d. URL https://openreview.net/forum?id=NTAhi2JEEE.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing*

Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022, 2022a. URL http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022b. URL http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

T. Wei, N. Sachdeva, B. Coleman, Z. He, Y. Bei, X. Ning, M. Ai, Y. Li, J. He, E. H. Chi, et al. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *ArXiv preprint*, abs/2511.20857, 2025. URL https://arxiv.org/abs/2511.20857.

C. Wu, Z. R. Tam, C. Lin, Y. Chen, and H. Lee. Streambench: Towards benchmarking continuous improvement of language agents. In A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/c189915371c4474fe9789be3728113fc-Abstract-Datasets_and_Benchmarks_Track.html.

W. Wu, K. Zhou, R. Yuan, V. Yu, S. Wang, Z. Hu, and B. Huang. Auto-scaling continuous memory for gui agent. *ArXiv preprint*, abs/2510.09038, 2025a. URL https://arxiv.org/abs/2510.09038.

Y. Wu, S. Liang, C. Zhang, Y. Wang, Y. Zhang, H. Guo, R. Tang, and Y. Liu. From human memory to ai memory: A survey on memory mechanisms in the era of llms. *ArXiv preprint*, abs/2504.15965, 2025b. URL https://arxiv.org/abs/2504.15965.

P. Xia, J. Chen, H. Wang, J. Liu, K. Zeng, Y. Wang, S. Han, Y. Zhou, X. Zhao, H. Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *ArXiv preprint*, abs/2602.08234, 2026. URL https://arxiv.org/abs/2602.08234.

W. Xu, Z. Liang, K. Mei, H. Gao, J. Tan, and Y. Zhang. A-mem: Agentic memory for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=FiM0M8gcct.

S. Yan, X. Yang, Z. Huang, E. Nie, Z. Ding, Z. Li, X. Ma, H. Schütze, V. Tresp, and Y. Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *ArXiv preprint*, abs/2508.19828, 2025. URL https://arxiv.org/abs/2508.19828.

L. Yang, Z. Yu, T. Zhang, S. Cao, M. Xu, W. Zhang, J. E. Gonzalez, and B. Cui. Buffer of thoughts: Thought-augmented reasoning with large language models. In A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/cde328b7bf6358f5ebb91fe9c539745e-Abstract-Conference.html.

Y. Yang, J. Li, Q. Pan, B. Zhan, Y. Cai, L. Du, J. Zhou, K. Chen, Q. Chen, X. Li, B. Zhang, and L. He. Autoskill: Experience-driven lifelong learning via skill self-evolution. *ArXiv preprint*, abs/2603.01145, 2026. URL https://arxiv.org/abs/2603.01145.

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

S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=WE_vluYUL-X.

Y. Ye, H. Jiang, F. Jiang, T. Lan, Y. Du, B. Fu, X. Shi, Q. Jia, L. Wang, and W. Luo. UMEM: unified memory extraction and management framework for generalizable memory. *ArXiv preprint*, abs/2602.10652, 2026. URL https://arxiv.org/abs/2602.10652.

H. Yu, T. Chen, J. Feng, J. Chen, W. Dai, Q. Yu, Y.-Q. Zhang, W.-Y. Ma, J. Liu, M. Wang, and H. Zhou. Memagent: Reshaping long-context LLM with multi-conv RL-based memory agent. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=k5nIOvYGCL.

C. Zhang, Y. Jian, Z. Ouyang, and S. Vosoughi. Working memory identifies reasoning limits in language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16896–16922, 2024.

H. Zhang, Q. Long, J. Bao, T. Feng, W. Zhang, H. Yue, and W. Wang. Memskill: Learning and evolving memory skills for self-evolving agents. *ArXiv preprint*, abs/2602.02474, 2026a. URL https://arxiv.org/abs/2602.02474.

S. Zhang, J. Wang, R. Zhou, J. Liao, Y. Feng, Z. Li, Y. Zheng, W. Zhang, Y. Wen, Z. Li, et al. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *ArXiv preprint*, abs/2601.03192, 2026b. URL https://arxiv.org/abs/2601.03192.

Y. Zhang, J. Shu, Y. Ma, X. Lin, S. Wu, and J. Sang. Memory as action: Autonomous context curation for long-horizon agentic tasks. *ArXiv preprint*, abs/2510.12635, 2025a. URL https://arxiv.org/abs/2510.12635.

Z. Zhang, Q. Dai, R. Li, X. Bo, X. Chen, and Z. Dong. Learn to memorize: Optimizing llm-based agents with adaptive memory framework. *ArXiv preprint*, abs/2508.16629, 2025b. URL https://arxiv.org/abs/2508.16629.

A. Zhao, D. Huang, Q. Xu, M. Lin, Y. Liu, and G. Huang. Expel: LLM agents are experiential learners. In M. J. Wooldridge, J. G. Dy, and S. Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 19632–19642. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29936. URL https://doi.org/10.1609/aaai.v38i17.29936.

B. Zheng, M. Y. Fatemi, X. Jin, Z. Z. Wang, A. Gandhi, Y. Song, Y. Gu, J. Srinivasa, G. Liu, G. Neubig, and Y. Su. Skillweaver: Web agents can self-improve by discovering and honing skills. *ArXiv preprint*, abs/2504.07079, 2025. URL https://arxiv.org/abs/2504.07079.

L. Zheng, R. Wang, X. Wang, and B. An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Pc8AU1aF5e.

H. Zhou, Y. Chen, S. Guo, X. Yan, K. H. Lee, Z. Wang, K. Y. Lee, G. Zhang, K. Shao, L. Yang, and J. Wang. Memento: Fine-tuning LLM agents without fine-tuning llms. *ArXiv preprint*, abs/2508.16153, 2025. URL https://arxiv.org/abs/2508.16153.

Z. Zhou, A. Qu, Z. Wu, S. Kim, A. Prakash, D. Rus, B. K. H. Low, and P. P. Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=XY8AaxDSLb.

Contents of Appendix

A. Prompts

In this section, we provide the full prompt templates used throughout different phases and components of our framework.

A.1. Prompt for Skill Curator

The following prompt templates demonstrate the input to the skill curator during training processes.

System Instruction

Role
You are an expert with a sophisticated skills curator. Our overall goal is to accomplish agent tasks. Your primary task is to convert past experiences of agent task execution into reusable, general skills, so that they can benefit and inspire future tasks.

Input Data
1. **Task Description**: The task to be accomplished.
2. **Past Skills**: A list of previously stored relevant skills, each with a skill name (identifier) and content.
3. **Agent Trajectory**: The step-by-step execution trace. Given the task, the agent interacts with the environment by selecting and calling specific past skills. This trajectory captures the sequence of skill invocations and the resulting transitions used to pursue the goal.
4. **Result**: Whether the agent successfully completed the task or not.

Critical Constraints:
- **Skill Format**: Extract and store important information as skills using following Markdown format **strictly**.
- **No Specifics**: Avoid problem-specific details. Remove specific numbers/names. Replace with variables/concepts.
- **No Hallucination**: Do not invent facts.
- Each skill must be **Atomic, modular, and reusable**.

Skill Markdown Format and Content Instructions:
- **YAML Frontmatter (MANDATORY)**: Each skill MUST start with a YAML frontmatter block delimited by `---`. The YAML block MUST contain exactly two keys: `name` and `description`.
  - **Example Structure**:
  ---
  name: <Human-readable skill name>
  description: <One-sentence what/when/why/how summary, concise and actionable, this will be used for future references>

- **Markdown Body**: Immediately after the second `---`, provide instructions using Markdown headings.
  - Suggested sections: `# Workflow`, `# When NOT to use`. These headings are just examples, you can come up with more ideas; use and craft what's appropriate for clarity.
  - Ensure the content is atomic, general, and devoid of specific instance IDs.

Action Guidelines
1. Analyze the agent trajectory and its result. Identify what went well and what didn't.
2. If the trajectory is correct, extract reusable knowledge or skills. If the trajectory is incorrect, identify the failure point and extract skills that can help fix the issue.
3. Compare the extracted skills with past skills. Determine whether to **insert a new skill**, **update an existing skill**, or **delete an existing skill** using the following tools.

{Tool Signatures}

Input

Task Context \n ## Task Description \n {task_description} ## Past Skills \n {past_skills} ## Agent Trajectory\n {agent_trajectory} ## Result\n {result}

Figure 7 | System prompt used for skill curator during training process.

A.2. Prompt for Agent Executor

The following prompts are used for the frozen agent executor. These templates provide the agent with the current task description, a history of previous interactions, and a set of retrieved skills to guide its decision-making process. All prompts explicitly force chain-of-thought (CoT) (Wei et al., 2022b) reasoning.

For agent tasks including ALFWorld and WebShop, we follow GiGPO (Feng et al., 2025) and leverage its environment and prompt setting for inference.

new_skill_insert: If there is no existing relevant skill, create new skill with desired skill name and content.
Parameters: {"type": "object", "properties": {"skill_name": {"type": "string", "description": "The name of the new skill to create."}, "content": {"type": "string", "description": "The markdown content for the new skill."}}, "required": ["skill_name", "content"]}
Format the arguments as a JSON object.

---

skill_update: If the existing skill can be improved, update the specific skill by its <skill_name>.
Parameters: {"type": "object", "properties": {"skill_name": {"type": "string", "description": "The name of the skill to update. Skill name must exist and exactly match the title of an existing skill."}, "new_name": {"type": "string", "description": "The new skill name for the skill, which replaces the old name. If not provided, the skill name will remain unchanged."}, "new_content": {"type": "string", "description": "The new content for the skill, which will replace the entire old content. Please ensure full content if provided. If not provided, the skill content will remain unchanged."}}, "required": ["skill_name"]}
Format the arguments as a JSON object.

---

skill_delete: Delete an existing skill by its title.
Parameters: {"type": "object", "properties": {"skill_name": {"type": "string", "description": "The name of the skill to delete."}}, "required": ["skill_name"]}
Format the arguments as a JSON object.

Figure 8 | Tool call definition/signature of skill curator in Figure 7.

You are an expert agent operating in the ALFRED Embodied Environment. Your task is to: {task_description}

Past Relevant Skills

{retrieved_skills}

Current Progress

Prior to this step, you have already taken {step_count} step(s). Below are the most recent {history_length} observations and the corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {current_observation}
Your admissible actions of the current situation are: {admissible_actions}

Now it's your turn to take an action.
You should first reason step-by-step about the current situation with the help of past relevant skills. This reasoning process **MUST** be enclosed within **<think> </think>** tags.
Once you've finished your reasoning, you should choose an admissible action for current step and MUST present it within **<action> </action>** tags.

Figure 9 | Prompt for ALFWorld agent execution with relevant retrieved skills.

You are a reasoning expert with access to a list of skills. Use the skills below to provide correct responses to user queries.

Past Relevant Skills

{retrieved_skills}

Problem

{question}

Please reason step by step, using the past relevant skills when helpful, and put your final answer within \boxed{}.

Figure 11 | Prompt for agent execution in reasoning tasks with relevant retrieved skills.

A.3. Prompt Used During Training

During the RL training process, a reward r^{cnt} is assigned based on an external judge of Qwen3-32B to judge whether the curated skills are semantically meaningful and are likely to be useful for future tasks. We show the prompt to the external judge here.

You are an expert agent operating in the WebShop e-commerce environment. Your task is to: {task_description}

Past Relevant Skills

{retrieved_skills}

Current Progress

Prior to this step, you have already taken {step_count} step(s). Below are the most recent {history_length} observations and the corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {current_observation}
Your admissible actions of the current situation are: {admissible_actions}

Now it's your turn to take an action.
You should first reason step-by-step about the current situation with the help of past relevant skills. This reasoning process **MUST** be enclosed within **<think> </think>** tags.
Once you've finished your reasoning, you should choose an admissible action for current step and MUST present it within **<action> </action>** tags.

Figure 10 | Prompt for WebShop agent execution with relevant retrieved skills.

System Instruction

You are an expert memory analyst. Analyze the quality of the following content of skills memory based on the following criteria:

1. **ABSTRACTION:** The skill captures generalizable procedures or insights, not verbatim copies of the trajectory. Specific IDs, numbers, object names from the task have been replaced with variables or general concepts.
2. **REUSABILITY:** The skill is atomic and modular — it describes one coherent capability that could plausibly be triggered by future related tasks, rather than bundling unrelated steps.
3. **ACTIONABILITY:** The Markdown body provides concrete guidance (workflow, conditions, when-not-to-use) that an executor can act on, rather than vague advice.
4. **FAITHFULNESS:** All claims in the skill are supported by the trajectory; no fabricated facts, tools, or environment behaviors.

Respond ONLY with a JSON code block in this exact format:

```json
{
  "VALID": true/false,
  "ISSUES": [list any problems found],
  "EXPLANATION": "brief explanation of the assessment"
}
```

Input

Analyze the following skill content: \n{content}

Figure 12 | Prompt for using an external judge to assign a reward score r^{cnt} for generated skill contents.

A.4. Prompt for LLM-as-a-Judge to Obtain Correctness Signals

We present the prompts used to obtain the self-judged correctness signal $\mathbb{1}_{\xi_t}$ for self-evolution via LLM-as-a-judge using the corresponding frozen agent executor as the backbone model in Figures 13, 14 for ALFWorld, reasoning, and WebShop tasks, respectively.

B. Implementation Details

B.1. Hyperparameters

We present the choices for all hyperparameters during both the training and inference processes in Table 4 for different tasks.

B.2. Grouping Training Instances

In this section, we detail the two-stage pipeline used to turn the raw training set $\mathcal{D} = \{x_i\}_{i=1}^{N}$ into the grouped training set $\mathcal{G} = \{G_j\}_{j=1}^{M}$ of Section 3.2.1. Stage 1 annotates each instance with a structured set of latent attributes via an LLM annotator (Sec. B.2.1). Stage 2 assembles groups of related tasks by retrieving, filtering, and ranking candidates under a semantic phrase-level similarity (Sec. B.2.2). For

System Instruction

You are an expert judge evaluating whether an embodied agent successfully completed a household task in a text-based simulator. Output a single JSON object and nothing else.
Task
You will be given (1) the task description the agent was asked to complete, and (2) the full interaction trace between the agent and the simulator. Determine whether the agent fully completed the task.
What "success" means
- The agent's actions must have produced the world state the task description specifies. Every condition stated in the task must hold at the end of the trace.
- If the task implies a transformation must occur before a final placement or interaction, the transformation must be evidenced in the trace before the final step.
- Credit only effects that the simulator's observations confirm. Do not credit effects that the agent merely declared, planned, or assumed. - Ignore the agent's own claims of completion; rely solely on the simulator's observation strings.
- A trace that ends with the agent stuck in a loop, exhausting its step budget, or repeatedly emitting invalid actions is a failure regardless of partial progress.

Strictness
- If the trace is ambiguous about whether every required condition is satisfied at the end, output success=false.
- Partial completion is failure. Either every condition holds or the trace is a failure.

Output
Output exactly one JSON object with these fields, and nothing else: {{ "success": <truelfalse>, "rationale": "<one or two sentences citing the specific observations that prove success or failure>", "evidence_step": <integer step index where success was confirmed, or -1 if failure> }}

Input

Inputs
Task description \n{task_description}
Trajectory
The trajectory alternates between simulator OBSERVATION and agent ACTION.\n{trajectory}

Figure 13 | Prompt for LLM-as-a-judge to obtain the correctness signal to the current trajectory in the ALFWorld benchmark.

training of single-turn reasoning tasks, we instantiate the pipeline on DeepMath-103K (He et al., 2026a), which provides both the raw problems x_i and a scalar difficulty score $d_i \in \mathbb{R}$ that is reused as a curriculum signal by Stage 2. For multi-turn agentic tasks, we leverage the default task type annotation for each benchmark (e.g., 6 task types in ALFWorld) as they naturally expose a discrete partition of tasks into families that share the same underlying skills, and we can use this partition directly in place of the annotated attribute set Z_i.

B.2.1. Stage 1: Latent Attribute Annotation

We implement the attribute set Z_i of each instance x_i as a tuple of five phrase-lists,

$$Z_i = \big(T_i,\ S_i,\ C_i,\ R_i,\ P_i\big),$$

where T_i is the list of high-level *topics*, S_i the required *skills or capabilities*, C_i the underlying *mathematical concepts or theorems*, R_i the applicable *heuristic strategies*, and P_i the *common pitfalls*. Each dimension is populated by a small set of short phrases (at most five words each). The annotator is instructed to: (i) emit standardized terminology rather than free-form rationales, (ii) omit any content specific to the question text or its final answer, and (iii) use as few phrases per dimension as necessary to characterize the task. We enforce the output schema via structured decoding with a fixed JSON response schema, and query Gemini-2.5-Pro with the highest thinking-budget configuration. The exact annotation instruction is reproduced in Figure 16.

B.2.2. Stage 2: Group Construction

Given $\{(x_i, Z_i, d_i)\}_{i=1}^{N}$, we construct each group $G_j = (x_{j,1}, \ldots, x_{j,n})$ by sampling a seed task and then iteratively appending related tasks. The core primitive is a pair sampler that, given a source x_s, returns an admissible successor x_t; longer groups are obtained by iterating this primitive with a growing exclusion set so that instances within a group remain distinct.

System Instruction

You are a rigorous reasoning problem judge. Your task is to determine whether a model's solution to a reasoning problem is correct.
Task
You will be given: 1. A reasoning problem. 2. A candidate solution, which contain long reasoning process. Your job is to judge the correctness of the candidate solution.
Rules
- The candidate is correct if its final answer is mathematically equivalent to the correct answer and its reasoning does not rely on invalid steps that accidentally lead to the right answer.
- Minor formatting differences are acceptable.
- Equivalent mathematical forms are acceptable.
- If the final answer is correct but the reasoning contains a serious conceptual error that invalidates the derivation, mark it as incorrect unless the final answer is independently and clearly justified later.
- If the problem asks for an exact value, approximation alone is insufficient unless justified by the problem.
- If the candidate refuses, gives no final answer, or only restates the problem, mark it as incorrect.
Protocol
1. Identify the problem's required output.
2. Extract the candidate's final answer.
3. Independently verify whether the candidate's answer satisfies the problem.
4. Check whether the candidate's reasoning supports the answer.
5. Ignore unnecessary verbosity, irrelevant exploration, or alternative attempts if the final chosen solution is clear and valid.
Output
Return your judgment in the following JSON format only:
{"verdict": "correct" or "incorrect", "reason": "A concise explanation of why the solution is correct or incorrect."}

Input

Inputs
Problem \n{problem}
Solution with reasoning process\n{solution}

Figure 14 | Prompt for LLM-as-a-judge to obtain the correctness signal for single-turn reasoning problems.

Phrase similarity. Because the annotated phrases come from a large open vocabulary (e.g., *"pigeonhole principle"* vs. *"counting argument"*), exact set overlap is unreliable. We therefore score the similarity between any two phrase lists A and B using a *soft-Jaccard* $\mathrm{SJ}_\tau(A, B)$ that combines exact matches with a greedy one-to-one matching between remaining phrases under a sentence-embedding cosine similarity (computed with `all-MiniLM-L6-v2` (Reimers and Gurevych, 2019)) above a threshold τ. We write $m_\tau(A, B)$ for the resulting integer *matched-pair count*, which we use alongside SJ_τ in the filters below.

Dependency gate. For a source x_s and candidate x_t, we accept the pair only when all of the following hold:

1. *Shared foundation:* $m_\tau(C_s, C_t) \geq \kappa_C$ and $m_\tau(S_s, S_t) \geq \kappa_S$;
2. *Shared reasoning:* $m_\tau(R_s, R_t) + m_\tau(P_s, P_t) \geq 1$;
3. *Not a near-duplicate:* $\mathrm{SJ}_\tau(T_s, T_t) \leq \theta_T$ and the weighted overall similarity $\Omega(x_s, x_t) \leq \sigma_{\max}$;
4. *Not too unrelated:* $\Omega(x_s, x_t) \geq \sigma_{\min}$;
5. *Progression:* x_t introduces at least one new concept or skill, i.e. $|C_t| > m_\tau(C_s, C_t)$ or $|S_t| > m_\tau(S_s, S_t)$;
6. *Curriculum direction:* $d_t - d_s \geq \delta_{\min}$.

Here Ω is a convex combination of per-dimension soft-Jaccard scores across $\{C, S, R, P, T\}$ with weights listed in Table 5. Conditions (1)–(2) ensure genuine reuse of foundational knowledge and reasoning machinery; (3)–(4) place the pair in a useful "related but not redundant" band; (5) guarantees that x_t carries something new for the skill curator to compress into the library; and (6) enforces a forward curriculum.

System Instruction
You are an expert judge evaluating whether a shopping agent purchased an item that matches a user's instruction in a web-shopping simulator. Output a single JSON object and nothing else. # Task You are given (1) the user's shopping instruction and (2) the agent's trajectory. Score how well the agent's purchase satisfies the instruction. **## How to score** The instruction encodes a product target, zero or more required attributes of that target, and possibly a price constraint. Decompose your evaluation into the following sub-scores, then average them into a single score in [0, 1]: 1. **Product type match**: 1 if the purchased product belongs to the category named in the instruction; otherwise 0. 2. **Attribute coverage**: the fraction of attributes explicitly named in the instruction that the purchased item (with its chosen options) is shown to satisfy. If the instruction names no attributes, score 1. 3. **Price constraint**: 1 if the purchase price satisfies the constraint stated in the instruction. If no price constraint is stated, score 1. 4. **Purchase completion**: 1 if the trajectory ends with a confirmed purchase action on a concrete product page; 0 otherwise. The final `score` is the mean of the four sub-scores. Define `success` as `score >= 0.5`. **## Strictness** - Award attribute credit only when the page text or the agent's selected options provide positive evidence; do not infer attributes from the absence of contradiction. - A purchase made on the wrong product type forces score = 0 regardless of the other sub-scores. # Output Output exactly one JSON object and nothing else: {{ "subscores": {{ "product_type": < 0 | 1>, "attribute_coverage": <float in [0,1]>, "price": < 0 | 1>, "purchased": < 0 | 1> }}, "score": <float in [0,1], the mean of subscores>, "success": <truelfalse>, "rationale": "<one or two sentences>" }}"""
Input
Inputs ## User instruction {instruction} ## Trajectory The trajectory alternates between OBSERVATION and ACTION. Long observations may be truncated; the final observation is preserved in full so you can inspect the purchased item. \n{trajectory}

Figure 15 | Prompt for LLM-as-a-judge to obtain the correctness signal to the current trajectory for the WebShop benchmark.

Candidate retrieval and scoring. Scoring all $N-1$ alternatives per source is prohibitive, so we precompute an inverted index over the dependency fields $\{C, R, P\}$: for each source x_s, the candidate pool consists of tasks that share at least one exact dependency phrase with x_s, capped at K_{inv} entries via uniform subsampling. Routing retrieval through dependency fields rather than topics prevents groups from collapsing onto a single narrow subject. Among the candidates that pass the gate, we select the one that maximizes

$$s(x_s, x_t) \;=\; \sum_{f \in \{C,S,R,P,T\}} w_f \, \mathrm{SJ}_\tau(f_s, f_t) \;+\; \lambda \cdot b(d_s, d_t),$$

where $b(\cdot)$ is a bounded difficulty bonus that rewards moderate forward steps. If no inverted-index candidate passes the gate, we fall back to a uniform random pool of size F and re-apply the same gate and scoring; this catches pairs whose phrases agree semantically but not lexically. Extensions sourced from the fallback pool are tagged so downstream training can audit or downweight them. The difficulty gap $d_t - d_s$ is additionally modulated by a randomized curriculum mode $(p_\uparrow, p_=, p_\downarrow)$; for our main experiments, we use an almost exclusively forward curriculum, which produced a more stable training signal than mixed curricula.

Hyperparameters. Table 5 lists all hyperparameters of the Stage 2 pipeline and the values adopted for our main experiments. The weights were tuned on a held-out subset of 200 source tasks by manually inspecting sampled pairs for prerequisite quality; we found the pipeline largely insensitive to small perturbations of the weights but noticeably sensitive to the progression and overall-similarity-band conditions, removing either of which produced markedly more trivial or degenerate pairs.

Table 4 | Hyperparameters for SkillOS for training and inference settings.

Hyperparameter	Value		
	ALFWorld	WebShop	Reasoning
RL Training			
Learning rate		1×10^{-6}	
Batch size		32	
KL loss Coef		0.001	
Max Prompt Length		16,384	
Max Response Length		4,096	
GRPO group size		8	
Temperature		1.0	
Steps	60	50	100
Data Grouping Size	10	10	Random(5,12)
Agent Executor Inference			
Top-K skill retrieval		5	
Max number of turns	30	30	1
Action history length	3	3	-

B.3. Experiment Setup

B.3.1. Datasets

In this section, we provide a detailed introduction to all the datasets involved in this paper.

ALFWorld. ALFWorld (Shridhar et al., 2021) is a text-based interactive benchmark that aligns the TextWorld engine with the embodied ALFRED environment, enabling agents to learn high-level household policies through natural-language interaction. The benchmark covers six task types — Pick & Place, Examine in Light, Clean & Place, Heat & Place, Cool & Place, and Pick Two & Place — situated in 120 simulated rooms spanning kitchens, bedrooms, bathrooms, and living rooms. It provides 3,553 training tasks, together with 140 valid_seen tasks for the test set. At each step, the agent receives a textual description of its surroundings together with a goal instruction (e.g., "put a hot apple in the fridge") and must issue high-level commands such as go to, take, open, heat, and put.

WebShop WebShop (Yao et al., 2022) is a simulated e-commerce web environment designed to benchmark language agents on realistic, grounded shopping tasks. The environment is populated with 1.18 million real-world products scraped from Amazon and 12,087 crowd-sourced natural-language instructions, partitioned into 10,587 training, 1,000 dev, and 500 test instructions. Given an instruction (e.g., “I’m looking for a quick-release fitness strap band in teal, priced lower than $40.00”), the agent interacts with the environment via two action types — search[query] and click[button] — to locate and purchase a product that matches the specified attributes, type, options, and price. At the end of each episode, a programmatic reward in [0, 1] is computed by comparing the purchased item against the ground-truth product specification. Following the standard evaluation protocol used in prior LLM-agent work, we evaluate on the 500 held-out test instructions.

DeepMath-103K DeepMath-103K (He et al., 2026a) is a large-scale, decontaminated mathematical reasoning dataset containing approximately 103K problems at high difficulty (primarily AoPS Levels 5–9), spanning algebra, calculus, number theory, geometry, probability, and discrete mathematics.

You are an expert in data annotation and mathematical reasoning.
Given a mathematical question, generate one or more phrases (less than 5 words) that thoroughly and precisely describe the characteristics of the math problem in the following dimensions:
1. Topic
2. Skills or Capabilities
3. Math Concepts or Theorems
4. Heuristic Strategy
5. Common Pitfalls

Requirements
- The annotations should be phrases only, avoid lengthy sentences
- Do NOT include any context or specifics from the question or solution
- Put your response in JSON format.
- Use as less phrases as possible for each dimension
- Use standardized/acknowledged phrases/terminologies only since phrases generated will be used for large-scale data processing

Figure 16 | System instruction used to elicit Z_i from each task in $\mathcal{D}$.

Table 5 | Hyperparameters of the Stage 2 grouping pipeline.

Symbol	Meaning	Value
—	Phrase encoder	`all-MiniLM-L6-v2`
τ	Cosine threshold for fuzzy phrase matching	0.60
κ_C	Minimum matched concept pairs	1
κ_S	Minimum matched skill pairs	1
θ_T	Maximum topic soft-Jaccard	0.65
$\sigma_{\min}, \sigma_{\max}$	Overall-similarity band	0.30, 0.85
$\delta_{\min}$	Difficulty-delta floor	0.0
$(w_C, w_S, w_R, w_P, w_T)$	Dimension weights	(5, 4, 3, 1, 2)
λ	Difficulty-bonus weight	1.0
$(p_\uparrow, p_=, p_\downarrow)$	Mode probabilities	(0.80, 0.20, 0.00)
$[\Delta_{\min}, \Delta_{\max}]$	Gap in EASY→HARD mode	[0.5, 3.0]
$\Delta_=$	Maximum $\lvert d_t - d_s \rvert$ in SAME mode	0.3
K_{inv}	Inverted-index subsample cap	2,000
F	Fallback pool size	200

Each problem is paired with a verifiable final answer — enabling rule-based RL rewards — together with a difficulty score, topic label, and three DeepSeek-R1 (Guo et al., 2025) chain-of-thought solutions. Specifically, we annotate a subset with around 33,000 problems, with a final 20,000 set of grouped training instances.

AIME24 & AIME25. A collection of demanding mathematical problems sourced from the 2024 and 2025 American Invitational Mathematics Examination (AIME), with 30 problems each year. Problems encompass algebra, geometry, number theory, and combinatorics. Created to assess large language models' sophisticated mathematical reasoning abilities, the dataset presents substantial difficulty, systematic multi-phase solutions, and distinctive answers, establishing it as a robust benchmark for evaluating advanced analytical capabilities.

GPQA. Short for Graduate Level Google-Proof Q&A Benchmark (Rein et al., 2024), GPQA comprises a collection of demanding text-based multiple choice problems authored by subject specialists in biology, physics, and chemistry, intentionally crafted to be "exceptionally challenging". We use the "GPQA-Diamond" subset for testing, which has 198 problems in total.

B.3.2. Baselines

We compare SkillOS against five representative baselines that span memory-free agents, recent memory-augmented methods, and two internal variants of our own framework. All baselines share the same frozen Agent Executor and are evaluated under identical task suites, retrieval budgets, and decoding settings to isolate the contribution of the memory mechanism.

(i) No Memory. A memory-free baseline in which the Agent Executor solves each task independently, without access to any external memory or cross-task knowledge transfer. Each episode begins from a blank state, and no information is retained across tasks. This baseline establishes a lower bound and isolates the contribution of any form of accumulated experience.

(ii) ReasoningBank (Ouyang et al., 2026). A recent memory-augmented method that distills reusable reasoning insights from past trajectories and stores them as a searchable bank for future tasks. At inference time, relevant insights are retrieved and injected into the executor's context to guide reasoning. ReasoningBank represents the class of experience-distillation approaches, which emphasize the content of stored knowledge but rely on fixed, heuristic policies for deciding what to write or discard.

(iii) MemP (Fang et al., 2025b). A procedural-memory method that induces reusable procedures from agent experience and applies advanced memory-management strategies — including consolidation, forgetting, and re-indexing — to maintain the memory store over time. MemP represents the class of rule-based memory management approaches, which feature more sophisticated maintenance policies than ReasoningBank but still prescribe curation decisions through hand-designed heuristics rather than learning them from downstream task feedback.

(iv) SkillOS-base. A variant of our framework in which the Skill Curator is instantiated with the same open-source backbone as SkillOS but without any RL fine-tuning, while all other components remain identical to SkillOS. This baseline serves two purposes: (a) it provides a lower-bound reference point that reflects the intrinsic prompting-based curation ability of the open-source backbone prior to optimization, and (b) it isolates the contribution of our GRPO-based training, since SkillOS-base shares exactly the same model architecture, prompting template, and memory interface as SkillOS but forgoes end-to-end optimization against task performance.

(v) SkillOS-gemini. A variant of our framework in which the Skill Curator is instantiated with Gemini-2.5-Pro instead of a trained open-source model, while all other components remain identical to SkillOS. This baseline serves two purposes: (a) it provides a strong closed-source reference point for the upper bound of prompting-based curation, and (b) it isolates the effect of our GRPO-based training, since SkillOS-gemini shares the same prompting template and memory interface as SkillOS but forgoes RL optimization against task performance.

Together, these baselines cover the main design axes along which memory-augmented agents differ from SkillOS: whether memory exists at all (i), how stored knowledge is represented (ii vs. iii), and whether curation decisions are prescribed by heuristics or learned from task feedback (ii and iii vs. SkillOS), as well as whether the curator itself benefits from RL optimization (iv and v vs. SkillOS).

B.3.3. Evaluation Metrics

We evaluate SkillOS and all baselines along two complementary axes — **task effectiveness** and **action efficiency** — using metrics tailored to each benchmark. Across all benchmarks and methods, every configuration is run with three independent random seeds; we report the mean across seeds, with one standard deviation shown as a subscript (e.g., $85.7_{\pm 1.6}$). Within each backbone block of

Tables 1 and 2, the best value in each column is highlighted in **bold**.

Success Rate (SR ↑). Our primary effectiveness metric on both ALFWorld and WebShop. On ALFWorld, SR is the fraction of evaluation episodes in which the agent reaches the goal state within the step budget, yielding a binary $\{0, 1\}$ outcome per episode. We report SR both per task category — *Pick*, *Look*, *Clean*, *Heat*, *Cool*, and *Pick2* — and as a macro-average (*Avg. SR*) across the six categories, so that categories with fewer tasks are not dominated by larger ones. On WebShop, following (Yao et al., 2022), SR is the fraction of episodes whose final reward equals exactly 1, i.e., the purchased product fully matches all specified attributes, options, type, and price constraints.

WebShop Score (↑). In addition to SR, WebShop provides a dense per-episode reward in $[0, 100]$ that credits partial matches on attributes, options, type, and price even when the purchase is not a perfect match. We report the average score across evaluation episodes as a finer-grained complement to SR: two methods with similar SR may differ substantially in how close their near-misses are to the target product.

Number of Steps (Steps ↓). Our efficiency metric on ALFWorld and WebShop. *Steps* is the average number of environment actions the agent issues per episode, computed over all evaluation episodes regardless of success. Failed episodes contribute steps up to their termination point (task completion, max-step cutoff, or early stop). This metric captures a dimension that SR and Score alone cannot: two methods may achieve comparable effectiveness while differing substantially in how efficiently they reach the goal, which has direct implications for inference cost and deployment feasibility.

Accuracy (Acc. ↑) on reasoning benchmarks. For the single-turn reasoning datasets — AIME24, AIME25, and GPQA — we report exact-match accuracy: the fraction of questions whose extracted final answer matches the ground truth. For AIME24 and AIME25, we adopt the evaluation protocol from the HuggingFace `math_verify`[1] toolkit, which parses the model's final boxed expression and verifies mathematical equivalence to the reference answer (accounting for equivalent numerical forms, simplifications, and formatting variants). For GPQA, which is a multiple-choice benchmark, we extract the predicted option letter from the model's response and score it as correct if and only if it exactly matches the ground-truth option. We additionally report an average accuracy (*Avg. Acc.*) across the three datasets to summarize overall reasoning ability.

Evaluation protocol. All methods share the same frozen Agent Executor, retrieval budget (top-k skills retrieved via BM25), maximum step budget, and decoding temperature within each backbone, so that differences in the reported metrics are attributable to the memory mechanism rather than to confounding inference settings. Unless stated otherwise, all numbers in the main paper are computed on the official held-out evaluation splits of each benchmark.

C. Additional Analyses

C.1. Results on Gemini-3.1-Flash-Lite

In addition to the Qwen3-8B/32B and Gemini-2.5-Pro executors used in the main paper, we further evaluate SkillOS on ALFWorld with the more recent Gemini-3.1-Flash-Lite as the frozen Agent

[1] https://github.com/huggingface/Math-Verify

Table 6 | Experiment results on ALFWorld benchmark. Success rate (SR ↑) and the number of steps (Steps ↓) are reported on 6 subsets for Gemini-3.1-Flash-Lite as frozen executor.

Methods	Pick (35)	Look (13)	Clean (27)	Heat (16)	Cool (25)	Pick2 (24)	Avg. SR (140)	Steps
No Memory	85.7 $_{0.0}$	59.0 $_{8.9}$	67.9 $_{9.3}$	25.0 $_{6.2}$	38.7 $_{2.3}$	66.7 $_{0.0}$	61.2 $_{2.3}$	18.5
ReasoningBank	87.6 $_{4.4}$	71.8 $_{4.4}$	63.0 $_{0.0}$	**52.1** $_{14.4}$	48.0 $_{10.6}$	62.5 $_{0.0}$	66.0 $_{2.7}$	17.6
MemP	84.3 $_{6.1}$	57.7 $_{5.4}$	63.0 $_{0.0}$	28.1 $_{4.4}$	34.0 $_{2.8}$	62.5 $_{0.0}$	58.6 $_{1.0}$	19.3
SkillOS-base	86.7 $_{1.6}$	61.5 $_{0.0}$	66.7 $_{0.0}$	41.7 $_{6.2}$	38.7 $_{16.0}$	68.1 $_{2.4}$	63.6 $_{3.9}$	17.7
SkillOS-gemini	**96.2** $_{1.6}$	61.5 $_{13.3}$	74.1 $_{3.7}$	31.2 $_{12.5}$	66.7 $_{4.6}$	68.1 $_{2.4}$	71.2 $_{2.9}$	16.1
SkillOS	88.6 $_{0.0}$	**84.6** $_{13.3}$	**77.8** $_{0.0}$	37.5 $_{17.2}$	**68.0** $_{8.0}$	**68.1** $_{2.4}$	**73.1** $_{2.7}$	**15.5**

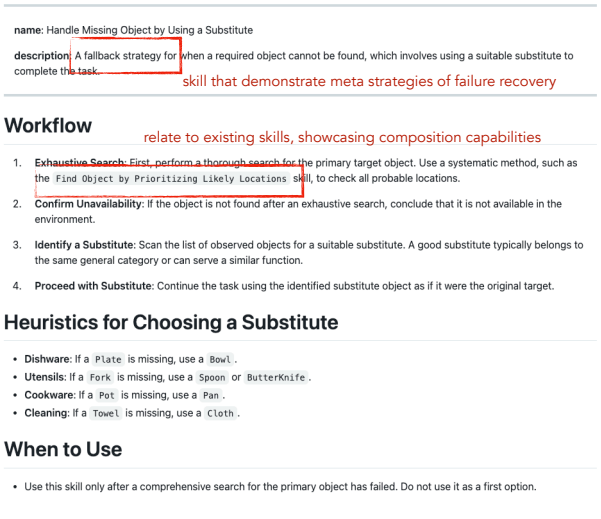
Executor, to verify that our gains generalize to newer model families. Results are reported in Table 6.

SkillOS achieves the highest average success rate (73.1%), outperforming the strongest external baseline ReasoningBank (66.0%) by **+7.1 points** and the No-Memory baseline (61.2%) by **+11.9 points**, while requiring the fewest interaction steps (15.5 vs. 18.5 for No Memory). The two internal variants reproduce the ordering observed in the main experiments: SkillOS-base reaches only 63.6% — barely above No Memory — confirming that the open-source backbone cannot recover the curation policy through prompting alone, and SkillOS-gemini improves to 71.2% but is still surpassed by SkillOS despite using a much stronger curator backbone. This reinforces our main finding that *learning* the curator with task-level feedback contributes more than scaling up the curator model. We also note that MemP (58.6%) underperforms even No Memory under this executor, suggesting that hand-designed curation heuristics are brittle when the executor is less capable, whereas the policy learned by SkillOS remains robust. Per-subset, SkillOS wins on four of six subsets, with particularly large margins on *Look* (84.6% vs. 71.8%) and *Cool* (68.0% vs. 48.0%); the remaining two subsets are won by SkillOS-gemini (*Pick*) and ReasoningBank (*Heat*), on which SkillOS nonetheless remains competitive. Overall, these results confirm that the advantage of SkillOS transfers cleanly to a newer executor family.

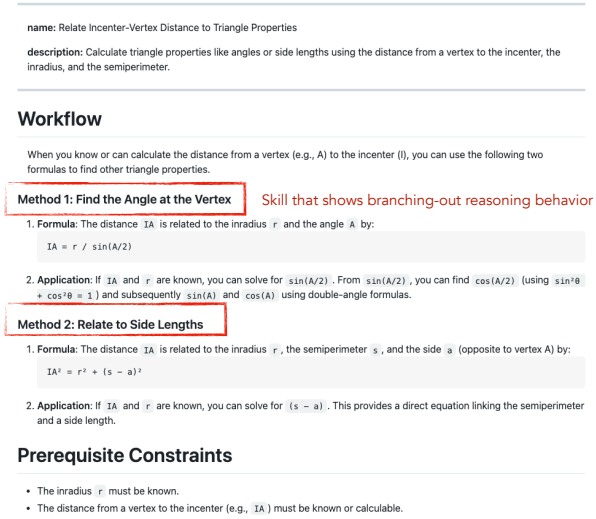
C.2. Case Studies

Curated Skills for Different Tasks. Figure 17 presents two representative skills curated by SkillOS that illustrate qualitatively different curation patterns across task types. For agentic tasks (Figure 17(a)), the curator distills a meta-strategy for failure recovery: rather than memorizing a specific object-search trajectory, it abstracts the recovery procedure into a reusable workflow (*exhaustive search → confirm unavailability → identify a substitute → proceed with substitute*) and explicitly references existing skills, demonstrating compositional curation. For reasoning tasks (Figure 17(b)), the curator captures *branching-out reasoning*: a single skill on inradius–circumradius–semiperimeter relations encodes multiple solution paths (relating the target distance to either the in/circumradius or the side lengths), each paired with its formula, application, and prerequisite constraints. Together, these examples show that SkillOS learns to produce skills tailored to the structure of the underlying task: procedural and composable for agentic settings, and multi-path with explicit preconditions for reasoning settings, rather than verbatim trajectory copies.

How SkillOS Curates Better Skills Compared to Baselines. We further qualitatively compare the skills curated by SkillOS against those produced by the baseline curator. In the math-reasoning case as shown in Figure 18, SkillOS-base outputs only a generic high-level recipe based on partitioning into disjoint sets, without explicit formulas, constraints, or examples. By comparison, SkillOS



(a) Skill curated by SkillOS for agentic tasks



(b) Skill curated by SkillOS for reasoning tasks

Figure 17 | Case studies of curated skills by SkillOS.

curates a much more useful skill that provides a concrete counting framework, including explicit constraint formulation, equation setup, and a worked example tailored to the target sub-problem. These examples show that RL-trained skill curation improves not only the correctness of the curated content, but also its specificity and usability, enabling skills to better capture the underlying structure of tasks.

How Curated Skills Help to Solve Tasks Successfully. Figure 19 illustrates a representative example of how curated skills improve agent behavior in interactive environments. Given the task "look at the CD under the desklamp," the memory-free baseline fails to infer the correct object–location relation and performs an inefficient search over irrelevant containers, eventually exhausting the step budget. In contrast, SkillOS retrieves a skill that encourages the agent to examine objects under or around light sources when the instruction refers to an object being "under" a lamp. Guided by this reusable strategy, the agent first locates and picks up the CD near the desk area, then moves to the desklamp and inspects the correct target location, completing the task successfully. This case highlights that curated skills do not merely memorize task-specific action sequences; instead, they provide transferable decision guidance that helps the agent focus exploration on semantically relevant objects and locations, reducing unnecessary interactions and improving task success.

D. Limitations

Retrieval Mechanism. Our current implementation relies on a relatively simple keyword-based retrieval mechanism, such as BM25, to retrieve relevant skills from the skill repository. This design choice allows us to isolate the main focus of this work: studying how skills can be curated, updated, and organized through experience-driven learning. However, more advanced retrieval methods, such as dense retrieval, hybrid retrieval, or learned retrievers, may further improve the relevance of retrieved skills and thus lead to stronger downstream performance. We leave the joint optimization of skill curation and skill retrieval to future work.

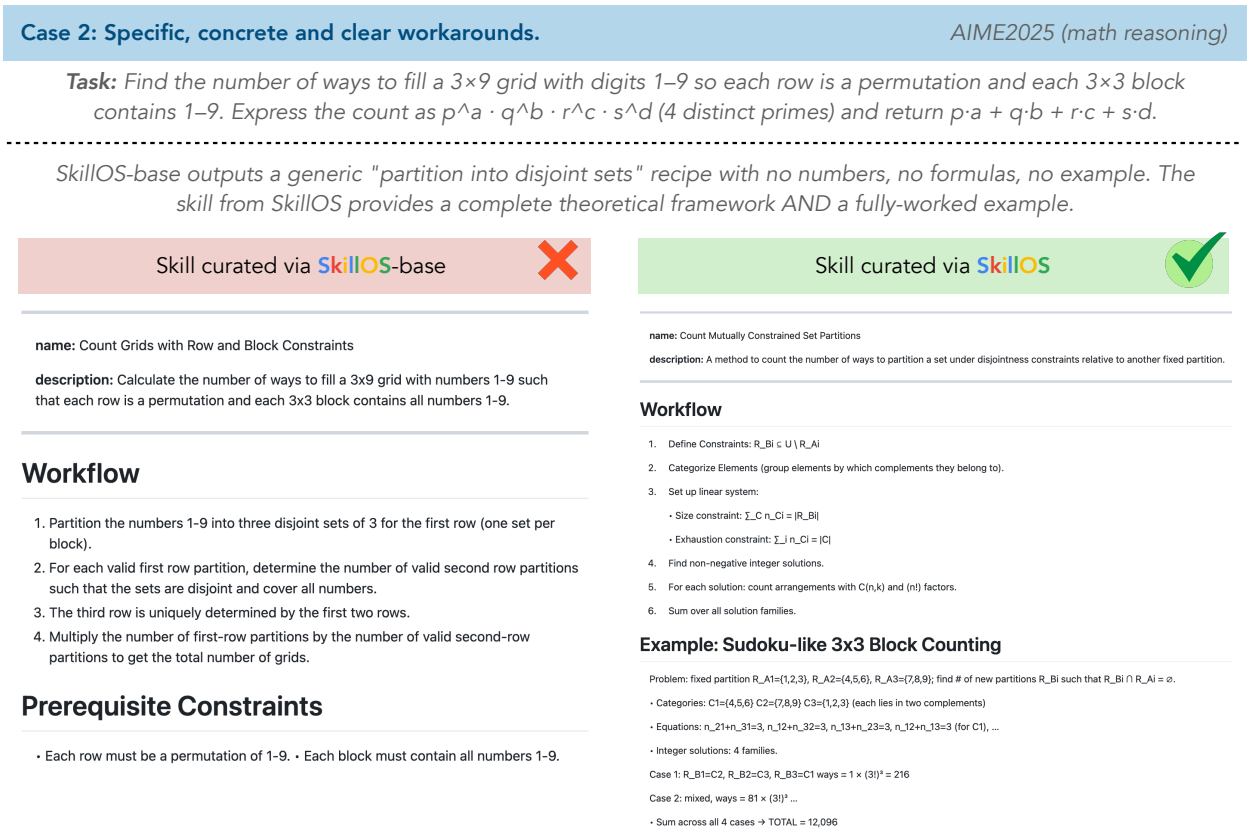


Figure 18 | Case study on math-reasoning skill curation. SkillOS-base produces a generic partitioning recipe, while SkillOS curates a concrete and reusable counting framework with explicit constraints, equations, and a worked example.

Simplified Skill Representation. Following Anthropic's skill paradigm (Anthropic, 2025b), we instantiate each skill as a single Markdown file that combines a YAML frontmatter and Markdown body. This simplification keeps the curator's action space tractable, but it discards two affordances of the original SKILL.md format: (i) supporting scripts and external resource files that allow skills to encapsulate executable procedures rather than purely declarative knowledge, and (ii) hierarchical organization in which a top-level skill can reference or compose lower-level sub-skills. As a result, behaviors that are most naturally expressed as runnable code or as compositions of finer-grained primitives must currently be flattened into prose. Extending SkillOS to multi-file, hierarchical, and partially executable skills is a natural next step.

Frozen Agent Executor. Throughout training, we keep the agent executor $\pi_{\mathcal{L}}$ frozen and optimize only the skill curator $\pi_{\mathcal{S}}$. This decoupling is deliberate: it isolates the contribution of skill curation, makes the recipe modular across executors, and avoids confounding our analysis with executor-side adaptation. The downside is that the curator can only shape the system's behavior through what it writes into SkillRepo; any miscalibration between the curated skills and the executor's idiosyncrasies must be absorbed by the curator alone. Joint or alternating optimization of $\pi_{\mathcal{S}}$ and $\pi_{\mathcal{L}}$ may yield a better-aligned pair, at the cost of executor specificity and substantially higher training cost.

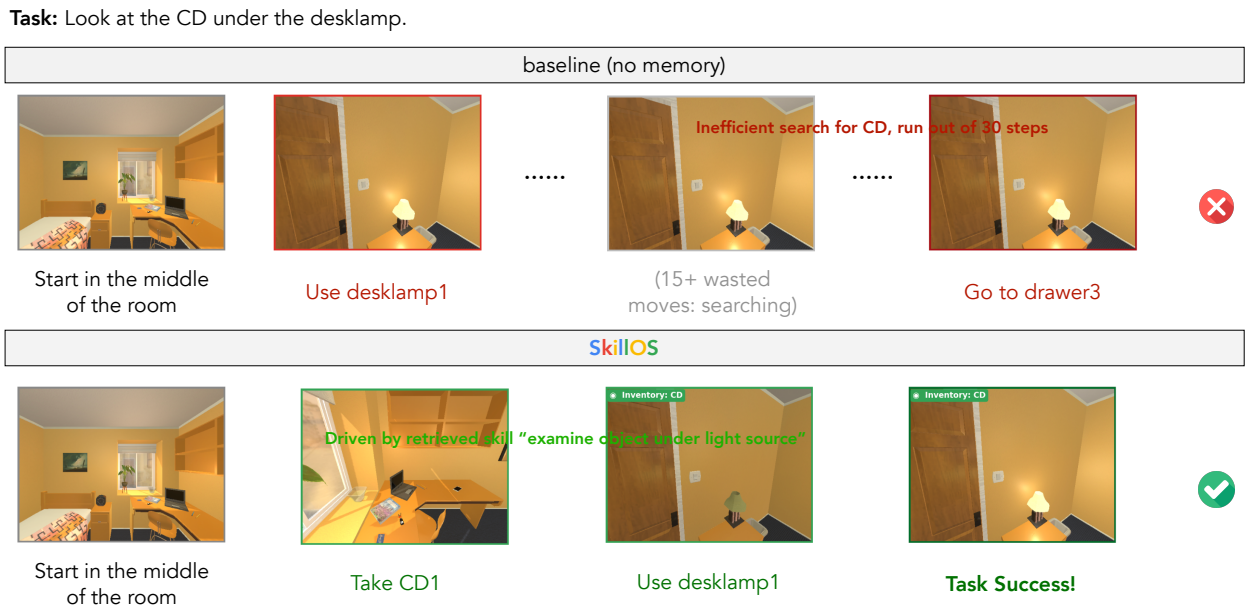


Figure 19 | Case studies of how skills curated by SkillOS successfully helped to solve a task in ALFWorld.

E. Future Research Directions

Our work opens several promising directions for future research.

Agentic Search over Experiential Memory. SkillOS currently retrieves relevant skills from SkillRepo through a fixed top-k BM25 lookup, treating retrieval as a static, one-shot operation. As the skill repository grows across thousands of tasks and domains, the bottleneck of self-evolving agents shifts from *what to store* to *how to reliably retrieve and inject the right fragments* at each decision step. A natural next step is to replace static retrieval with **agentic search**: letting the Skill Curator (or a dedicated retrieval agent) actively issue multiple queries, reformulate them based on intermediate evidence, and iteratively decide which skills to surface, cite, or compose for the executor. This reframes memory access as a first-class decision in the agent's policy rather than a preprocessing step, and opens the door to scaling SkillOS to memory stores orders of magnitude larger than those considered here.

Hierarchical and Compositional Skills. Our current skills are flat Markdown entries, each describing a single reusable pattern. Real agent competence, however, is hierarchical: high-level procedures invoke lower-level sub-skills, which in turn depend on primitive operations. Extending SkillRepo to support **hierarchical decomposition** — where the curator learns not only to insert, update, and delete skills but also to link, compose, and abstract them — could enable the agent to build increasingly expressive procedural libraries over time. This direction connects naturally to program-synthesis and library-learning literature, and would allow SkillOS to scale to longer-horizon tasks where single-skill retrieval is insufficient.

Multi-Agent and Shared Memory. SkillOS treats memory as a single agent's private artifact. In many realistic deployments, however, multiple agents operate in parallel (e.g., code review, multi-hop

research, collaborative robotics) and could benefit from **shared experiential memory**. Open questions include how to arbitrate conflicting curation decisions from different agents, how to attribute credit when a shared skill contributes to one agent's success but another's failure, and how to preserve specialization while enabling cross-agent transfer. Our GRPO-based curator provides a natural starting point, but extending it to the multi-agent credit-assignment setting is non-trivial and likely to require new algorithmic ideas.

F. Use of LLMs

We used LLMs as a general-purpose writing assist tool during the preparation of this submission. Specifically, LLMs were employed for polishing the clarity and readability of text (e.g., refining sentence structure, improving grammar, and shortening overly verbose phrasing). All research ideas, methodology design, experiments, analyses, and final writing decisions were conceived, implemented, and validated solely by the authors.