SKILLSANDBOX: SKILL VERIFICATION VIA DYNAMIC SCENARIO SYNTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-evolving agents distill task-solving experience into skills for future reuse, but these skills can encode incorrect procedures or non-transferable knowledge. It is therefore critical to verify each skill's *reusability*: whether its guidance remains useful beyond the experience from which it was distilled. Such verification requires observing how a skill affects execution in new tasks, yet existing tasks may not expose the situations where the target skill can actually be exercised. To construct such situations, we propose SKILLSANDBOX, a framework that dynamically synthesizes a task and its environment for each skill that are skill-relevant yet novel. A PROPOSER specifies the conditions to preserve and the source-specific details to vary, a BUILDER constructs an executable scenario, and a VERIFIER compares executions with and without the skill. The VERIFIER assesses executability, utility, and efficiency to assign a Keep or Reject verdict, determining whether the skill enters the library. Across ALFWorld and WebShop with three models, our framework consistently yields the strongest downstream performance and improved execution efficiency. Further analyses examine whether these gains reflect accurate assessment of skill reusability and identify which components of our verification framework contribute to them. [CODE]

1 INTRODUCTION

Self-evolving agents seek to improve their capabilities by learning from their own experiences (ang Gao et al., 2026; Yang et al., 2026). They distill task-solving experience into skills (Anthropic, 2025) that provide procedural guidance and store them in a library for future reuse (Wang et al., 2023; Ouyang et al., 2026b). However, distilled skills do not always guarantee the performance improvement (Li et al., 2026; Han et al., 2026; Gautam et al., 2026). Because task-solving experience contains redundant exploration, mistakes, and task-specific details, distillation may yield incorrect procedures or non-transferable knowledge (Ni et al., 2026; Ma et al., 2026; Xiong et al., 2026). Once stored and retrieved, such skills can repeatedly interfere with the agent's behavior (Dong et al., 2026; Liang et al., 2026). Self-evolving agents therefore need to verify each skill's *reusability*: whether the guidance remains useful when applied beyond the source experience.

Such verification requires execution evidence from a scenario that is both skill-relevant and novel. That is, the agent should actually encounter the situation the skill addresses in a new environment, so that the evidence reveals whether the guidance improves its behavior beyond the source experience. Because different skills address different situations, the scenario that provides this evidence is inherently skill-specific. However, existing approaches (Wang et al., 2025a; Zhang et al., 2026; Ouyang et al., 2026a) verify skills against a static pool of tasks that already exists, and merely select from it. Replaying the source task likely exposes the targeted situation, but only in the context the skill was distilled from, so the evidence is relevant but not novel (Chen et al., 2026a; Liu et al., 2026). Evaluating the skill on other existing tasks introduces new contexts, but since these tasks are defined independently of the skill, they need not expose the targeted situation.

Verification should therefore shift from *selecting* existing tasks, to *constructing* the task and environment around the target skill, on demand. To this end, we propose SKILLSANDBOX, a framework that dynamically synthesizes a task and environment where the target situation arises in a context different from the source experience to verify skills. Each synthesized task and its configured environment form a verification scenario, which serves as a controlled test of that skill. SKILLSANDBOX

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

consists of a PROPOSER, a BUILDER, and a VERIFIER, which design, construct, and evaluate this test, respectively. (i) The PROPOSER specifies the conditions that reproduce this situation and the modifications that distinguish the scenario from the source experience. (ii) The BUILDER realizes this specification as an executable task and its configured environment. (iii) The VERIFIER compares executions in the scenario with and without the skill and measures the effect of the skill in terms of executability, task success, and efficiency. Consequently, a reusable skill yields a positive effect across its scenarios and is admitted to the library, whereas a skill that is incorrect, overly specific, or not actionable yields no benefit or degrades execution and is rejected.

We evaluate SKILLSANDBOX on ALFWorld and WebShop by comparing the library of verified skills with libraries constructed by baseline methods, which either accumulate every generated skill or continually update the library as experience accumulates. Across three models, the verified library achieves the highest success rate with the fewest steps on both benchmarks. Further analyses show that the performance gains originate from the synthesized scenarios, as verification on source tasks or randomly selected tasks yields smaller gains and the gap widens on task types unseen in the source experience. In addition, the verification predicts the helpful and harmful effects of individual skills on held-out tasks. Each criterion also proves necessary, as the full VERIFIER outperforms variants that omit executability or efficiency.

Our contributions are as follows:

- We identify that skill verification should shift from *selecting* existing tasks to *constructing* the task and environment around the target skill on demand.
- We propose SKILLSANDBOX, which synthesizes a task and environment that expose the situation needed to verify the skill. Its PROPOSER and BUILDER generate scenarios, and VERIFIER measures the reusability of skill and determines whether to keep or reject it.
- We demonstrate on ALFWorld and WebShop that the skill library verified by ours improves downstream performance. Extensive analyses further validate the effect of SKILLSANDBOX and show that the synthesized scenarios provide a reliable verification environment.

2 PRELIMINARY ANALYSES: SKILL-SPECIFIC EVIDENCE IS SPARSE IN EXISTING EXPERIENCE

We first ask whether the scenario synthesis is actually necessary to verify skills. Can existing tasks provide the evidence needed for verification? To answer this question, we examine alternatives to scenario synthesis and evaluate whether either one reliably provides the skill-specific evidence.

- **Question I:** *Do semantically similar tasks expose the skill-relevant situation?*
- **Question II:** *How sparse are skill-relevant situations in downstream experience?*

2.1 QUESTION I: DO SEMANTICALLY SIMILAR TASKS EXPOSE THE SKILL-RELEVANT SITUATION?

We examine whether semantic similarity can identify existing tasks that expose the skill-relevant situation. We use 50 skills per benchmark, 140 held-out tasks in ALFWorld (Shridhar et al., 2020) and nested pools of 200, 300, and 500 tasks in WebShop (Yao et al., 2022a), keeping the same 50 skills across pool sizes. For each skill, we select the 10 held-out tasks whose instructions are most similar to the skill text. The agent executes each selected task with the target skill provided, and an LLM judge determines from the resulting trajectory whether execution encounters the skill-relevant situation.

Table 1: Mean skill-relevant tasks out of 10 retrieved tasks. N: candidate task count.

Executor	**ALFWorld**	**WebShop**		
N	140	200	300	500
Gemini 3.1 Flash	5.96	5.96	5.90	6.10
Qwen3.5-27B	6.00	6.60	6.58	6.66

Instruction similarity does not reliably identify tasks that exercise the target skill. As shown in Table 1, the skill-relevant situation is observed in only up to 6.66 out of 10 selected tasks per skill across the settings. Even expanding the candidate pool from 200 to 500 tasks yields only marginal

gains in skill-relevant coverage among the top-10 retrieved tasks. The main reason to this is that semantic similarity is measured at the task level, whereas skill relevance depends on what occurs during execution. Tasks with similar instructions can still induce different states, interactions, and intermediate decisions, and therefore need not expose the same skill-relevant situation. Consequently, task-level similarity does not reliably provide the execution evidence required for skill verification.

2.2 Question II: How sparse are skill-relevant situations in downstream experience?

A second alternative is to accumulate verification evidence from subsequent downstream experience (Zhang et al., 2026; Xiong et al., 2026; Ouyang et al., 2026a). We construct a library of 50 skills and stream 500 held-out tasks in a fixed random order, repeating each setting three times. For each task, referenced skill IDs are recorded in rationale and counted. We vary the number of skills accessible, with **Top-3** and **Top-5 Retrieval** providing the top-3 or 5 skills retrieved per task, respectively, and **Full Library** all 50 skills on every task. Figure 1 (a) reports the percentage of skills used in at least one task, and (b) reports the percentage used in at least 5 distinct tasks.

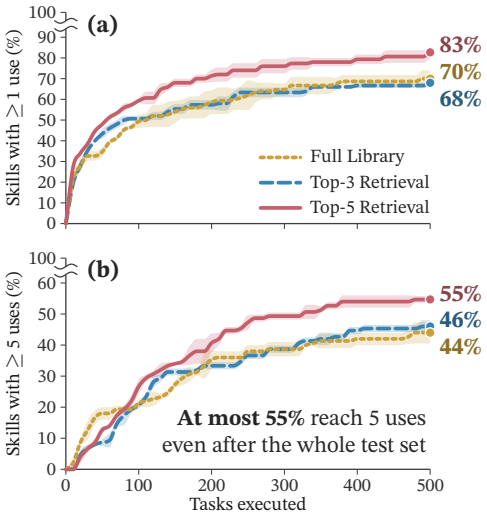


Figure 1: Percentage of skills used in (a) at least one task and (b) at least 5 tasks as downstream tasks are streamed and solved sequentially (Gemini 3.1 Flash-Lite, WebShop). Mean of three runs of the same task stream; bands show one standard deviation.

Skill-relevant situations remain sparse as downstream experience accumulates. (a) shows that coverage grows slowly after 300 tasks. Executing 200 additional tasks raises the percentage of skills used at least once by only 4.7–6.7 percentage points. After 500 tasks, only 68% of skills are used under Top-3 Retrieval, 83% under Top-5, and 70% under Full Library. Thus, depending on the access setting, 17–32% of skills remain *unexercised*, which means that their effects on the agent's behavior are never observed, leaving their reusability unverifiable. More reliable verification benefits from repeated evidence across distinct tasks, yet such opportunities are even more sparse. As shown in (b), only 44–55% of skills are used in at least 5 tasks. Even after 500 tasks, 45–56% of the skills fail to accumulate five execution opportunities. Overall, downstream experience leaves a substantial number of skills with insufficient evidence for verification.

Together, Questions I and II reveal complementary limitations of relying on existing tasks for verification. Neither reliably provides the skill-specific evidence that verification needs. We therefore construct verification scenarios that induce the target situation beyond its source experience.

3 Methodology

3.1 Problem Setup

We model an environment as $E = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, \kappa)$, where $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{O}$ are the state, action, and observation spaces, T is the state-transition rule, and the configuration κ specifies the environment's objects, arrangements, task-related data, and presentation of information. A task x specifies the goal and requirements. At each step, an agent π receives a partial observation (Kaelbling et al., 1998) $o_t \in \mathcal{O}$ of state s_t and takes an action $a_t \in \mathcal{A}$, yielding $s_{t+1} \sim T(\cdot \mid s_t, a_t)$, and the trajectory is $\tau_x = (o_0, a_0, o_1, a_1, \ldots)$. We call the environment and information encountered during execution the *task context*.

A synthesis specification δ programmatically specifies the requirements and changes that construct a new task x_δ and configuration κ_δ, which with the success criterion g_δ define

$$E_\delta = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, \kappa_\delta), \qquad \sigma_\delta = (x_\delta, E_\delta, g_\delta).$$

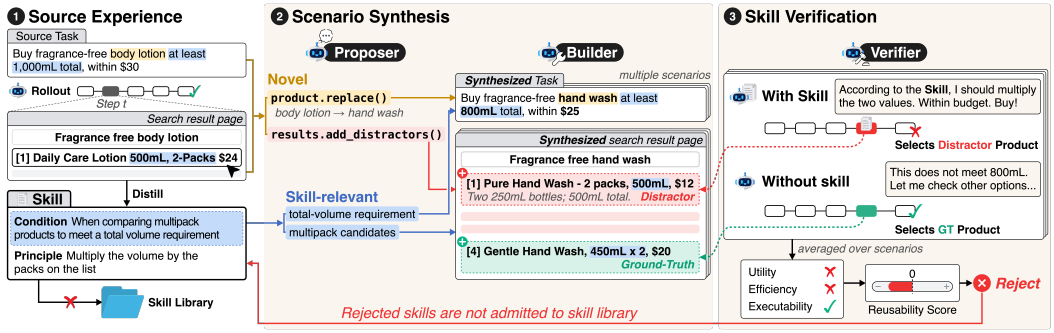


Figure 2: Overview pipeline of SKILLSANDBOX.

3.2 SKILL VERIFICATION

A skill $k = (\texttt{condition}, \texttt{principle})$ is distilled from a source task x_{src} and trajectory τ_{src}. The `condition` describes when the skill applies, and the `principle` provides knowledge or guidance for that situation. Verification assesses the skill's *reusability*, defined as its ability to guide executable behavior that improves task success or execution efficiency in new situations satisfying the `condition`. We verify the skill's reusability along three dimensions:

1. **Executability:** The `principle` can be executed through admissible actions in the environment when the `condition` is satisfied.
2. **Utility:** Applying the `principle` helps the agent achieve the task goal, improving task success relative to execution without the skill.
3. **Efficiency:** The `principle` enables the task completion with fewer actions than execution without the skill.

Evaluating these dimensions only on the source task x_{src} does not establish reusability in new task contexts, while arbitrary tasks may not reproduce the situation described by the `condition`. Verification scenarios must therefore be both *skill-relevant* and *novel*. A scenario is skill-relevant when completing the task requires addressing the situation described by `condition`, in which the guidance in `principle` can be evaluated. It is novel when the agent encounters a task context that differs from the source experience.

3.3 SCENARIO SYNTHESIS

To construct scenarios that are both skill-relevant and novel, we reproduce the situation described by the `condition` while reconstructing the new task context. We jointly construct a task x_δ and configuration κ_δ so that the `condition` holds during execution, while varying the details. Execution in the resulting scenario tests whether the `principle` remains useful when the situation described by the `condition` occurs in a context different from the source experience. Figure 2 illustrates this synthesis process. In SKILLSANDBOX, the PROPOSER specifies how to reproduce the condition in a new context, and the BUILDER constructs the scenario according to this specification.

PROPOSER. The PROPOSER first uses the `condition`, x_{src}, and τ_{src} to jointly plan reproduction and reconstruction. It encodes its natural language plan as a programmatic specification δ, using shared construction operations exposed by the construction interface $\mathcal{I}_E$. This interface specifies which task and configuration fields can be modified (e.g., `list_order`, `product_option`) and the operations permitted for each field (e.g., `add_distractor`, `remove`, `alter`). The specification includes variables for objects and values that the BUILDER later instantiates. For reproduction, it specifies the requirements that the task query and configuration must satisfy for the situation described by the `condition` to occur during execution. For reconstruction, it specifies the modifications that produce a context different from the source experience while satisfying the reproduction requirements. The resulting specification δ is expressed by the operations permitted by $\mathcal{I}_E$, providing the BUILDER with a plan for scenario construction.

BUILDER. To construct an executable scenario according to given δ, the BUILDER uses the task and configuration data $\mathcal{D}_E$ (e.g., `product_database` for WebShop, `object_list` for ALFWorld) available in E. The BUILDER first binds the variables in δ to concrete objects and values from $\mathcal{D}_E$. It then translates the specified operations into an executable construction program for E and runs it. As specified in δ, this program constructs new task instruction, environment, and corresponding ground truth, using $\mathcal{D}_E$.

To faithfully reproduce the skill's `condition` across diverse contexts, this construction proceeds within a validation loop. Using the same LLM model as the executor, the loop checks whether each constructed scenario satisfies the requirements in the PROPOSER's specification δ and whether it is identical to any previously generated scenario for the same skill. Scenarios that fail validation are discarded, and new candidates are generated until five valid scenarios are obtained. Together, x_δ and E_δ reproduce the situation described by the `condition` in a reconstructed task context, allowing the agent to reuse the `principle` beyond the source experience. The BUILDER returns the resulting verification scenario σ_δ:

$$\delta = \textsc{Proposer}(Skill, x_{\text{src}}, \tau_{\text{src}}, \mathcal{I}_E), \qquad \sigma_\delta = \textsc{Builder}_E(\delta; \mathcal{D}_E).$$

To illustrate what kind of scenarios are generated and how they are constructed, we provide modification statistics (Appendix C), and case studies (Appendix D).

VERIFIER. For each synthesized scenario, the VERIFIER compares trajectories τ^+ and τ^- produced by the same executor with and without skill k (Figure 2, right). At aligned steps with identical pre-action observations, Δu measures the difference in terminal success rewards discounted by the remaining action counts, capturing utility and efficiency. The executability indicator e is 1 when the action in τ^+ implements the skill's instruction and 0 otherwise. We average $e\,\Delta u$ over retained step pairs within each trajectory pair and then across scenarios to score reusability and assign a verdict (score calculation are detailed in Appendix B):

$$R(k) = \widehat{\mathbb{E}}[e\,\Delta u], \qquad \textsc{Verifier}(k) = \begin{cases} \textsc{Keep}, & R(k) > 0, \\ \textsc{Reject}, & R(k) \le 0 \end{cases}. \tag{1}$$

4 EXPERIMENTS

We first present the experimental setup (Section 4.1) and main results (Section 4.2). We compare SKILLSANDBOX with baselines that construct skill libraries with or without verification. To examine how effectively each method retains reusable skills, we build a library using each method and evaluate it on the same held-out tasks.

4.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate on ALFWorld (Shridhar et al., 2020) and WebShop (Yao et al., 2022a). ALFWorld provides six household task types, and WebShop requires agents to find and purchase products that satisfy user instructions.

Metrics. We report success rate (SR, %) on both benchmarks, average task score (Score, 0–1) on WebShop, and average steps across all episodes. To characterize how a skill library changes individual task outcomes beyond SR, we additionally report Recovery (Rec.) and Regression (Reg.) relative to the run without skill library of the same model. Recovery is the percentage of all held-out tasks that fail without skills but succeed with the library, whereas Regression is the percentage that succeed without skills but fail with the library.

Baselines. *No Skill* executes tasks without a skill library. *Vanilla Skill* retains all skills generated with the distillation prompt from SkillRL (Xia et al., 2026). We additionally compare with ReasoningBank (Ouyang et al., 2026b) and MemP (Fang et al., 2026b), as well as execution-based library curation methods including ExpeL (Zhao et al., 2024), ACE (Zhang et al., 2026), and SkillOS (Ouyang et al., 2026a). ExpeL, ACE, and SkillOS admit skills using execution feedback: ExpeL extracts insights from repeated attempts at the source task, whereas ACE and SkillOS update the library from subsequent tasks that are selected independently of any skill. Detailed descriptions of each baseline are provided in Appendix A. SKILLSANDBOX retains only the skills from Vanilla Skill that are verified as Keep.

Table 2: Main results on held-out ALFWorld and WebShop tasks. Bold marks the best value in each column for each model, and colored percentages are relative changes from No Skill. WebShop Score is reported on a 0–100 scale.

Verif.	Method	ALFWorld				WebShop				
		SR↑	Steps↓	Rec.↑	Reg.↓	SR↑	Score↑	Steps↓	Rec.↑	Reg.↓
Qwen3.5-9B										
–	No Skill	32.1	23.8	–	–	18.8	47.1	8.0	–	–
Unverif.	Vanilla Skill	57.9+80.4%	19.3	30.0	4.3	15.6-17.0%	38.0-19.3%	9.6	5.0	8.2
	ReasoningBank	53.6+67.0%	20.0	27.1	5.7	17.2-8.5%	38.4-18.6%	10.0	5.8	7.4
	MemP	42.9+33.6%	22.1	14.3	3.6	17.2-8.5%	41.2-12.6%	9.4	6.6	8.2
Src. task	ExpeL	42.1+31.2%	22.1	13.6	3.6	21.8+16.0%	52.9+12.4%	7.0	**8.2**	5.2
Dwnstr. task	ACE	45.0+40.2%	21.7	18.6	5.7	15.6-17.0%	35.4-24.9%	10.4	5.2	8.4
	SkillOS	40.0+24.6%	22.7	14.3	6.4	11.4-39.4%	25.2-46.6%	12.2	3.8	11.2
Synth. scenario	**SkillSandbox**	**66.4**+106.9%	**17.5**	**36.4**	**2.1**	**22.2**+18.1%	**59.1**+25.5%	**7.0**	8.0	**4.6**
Gemini 3.1 Flash-Lite										
–	No Skill	56.4	19.9	–	–	25.6	58.9	5.6	–	–
Unverif.	Vanilla Skill	49.3-12.6%	20.6	6.4	13.6	27.6+7.8%	54.9-6.8%	8.0	7.2	5.2
	ReasoningBank	52.9-6.2%	20.3	9.3	12.9	23.4-8.6%	53.7-8.9%	6.9	5.4	7.6
	MemP	55.0-2.5%	19.5	**12.1**	13.6	25.0-2.3%	55.3-6.1%	6.3	5.0	5.6
Src. task	ExpeL	50.0-11.3%	21.3	9.3	15.7	27.4+7.0%	59.4+0.8%	5.8	6.0	4.2
Dwnstr. task	ACE	42.9-23.9%	22.0	5.7	19.3	20.8-18.8%	41.5-29.5%	10.4	6.6	11.4
	SkillOS	51.4-8.9%	20.8	10.7	15.7	28.4+10.9%	56.4-4.3%	7.2	7.4	4.6
Synth. scenario	**SkillSandbox**	**60.0**+6.4%	**19.1**	**12.1**	**8.6**	**36.2**+41.4%	**64.8**+10.0%	**5.3**	**12.4**	**1.8**
Qwen3.5-27B										
–	No Skill	75.7	15.4	–	–	32.2	55.4	8.0	–	–
Unverif.	Vanilla Skill	75.0-0.9%	15.0	7.9	**8.6**	31.8-1.2%	54.5-1.7%	8.5	8.8	9.2
	ReasoningBank	67.1-11.4%	17.1	10.7	19.3	29.2-9.3%	53.0-4.4%	8.4	5.4	8.4
	MemP	52.9-30.1%	19.8	2.9	25.7	32.4+0.6%	57.5+3.8%	8.6	7.2	7.0
Src. task	ExpeL	70.0-7.5%	16.2	10.0	15.7	34.6+7.5%	54.7-1.2%	8.3	9.6	7.2
Dwnstr. task	ACE	74.3-1.8%	15.5	12.1	13.6	37.4+16.1%	59.7+7.9%	7.4	11.4	6.2
	SkillOS	63.6-16.0%	18.1	10.7	22.9	30.4-5.6%	49.7-10.3%	9.0	8.0	9.8
Synth. scenario	**SkillSandbox**	**80.0**+5.7%	**14.3**	**12.9**	**8.6**	**45.0**+39.8%	**62.2**+12.2%	**7.2**	**18.0**	**5.2**

Implementation details. We use Qwen3.5-9B (Qwen Team, 2026), Gemini 3.1 Flash-Lite (Google DeepMind, 2026), and Qwen3.5-27B (Qwen Team, 2026) as executors within the ReAct framework (Yao et al., 2022b). The same model is used for execution and skill generation, and our scenario synthesis and verification also use the same model. All baselines use the same 100 source tasks to execute and generate skills, and each library is frozen before held-out evaluation. We evaluate on 140 tasks from the official ALFWorld test sets and 500 tasks from the official WebShop test sets, retrieving the top-3 skills for each task (Appendix E).

4.2 Main Results

Unverified skills can degrade agent performance. As shown in Table 2, libraries admitted without our verification achieve lower SR than No Skill in most settings. This degradation is often accompanied by an increase in average steps, which means that the agent inefficiently spends more steps on redundant exploration yet solves fewer tasks. These results suggest that distilled skills can act as noise, impairing both task success and execution efficiency. Verifying their reusability before admission is therefore important for filtering out non-reusable skills.

Effective skill verification depends on the quality of execution evidence. SkillSandbox consistently outperforms execution-based curation methods (ExpeL, ACE, and SkillOS) across all models and benchmarks, improving SR over the strongest of these baselines by up to 21.4%p. These

results suggest that using execution feedback alone does not guarantee effective skill verification. ExpeL obtains evidence from repeated executions of the source task, whereas ACE and SkillOS use subsequent task stream. SKILLSANDBOX instead acquires the verification evidence from execution on the skill-specific scenarios. Section 5.1 validates the contribution of scenario synthesis by comparing with other execution-based evidence under the same rollout budget.

Reliable verification must account for both Recovery and Regression. On ALFWorld with Gemini, MemP and SKILLSANDBOX achieve the same Recovery, but SKILLSANDBOX has approximately 37% lower Regression. Consequently, MemP reduces SR relative to No Skill, whereas SKILLSANDBOX improves it. Reliable verification should therefore both help the agent solve previously failed tasks and preserve success on tasks it already solves.

Verification selects an effective subset of the original skill library. Compared with the unverified Vanilla Skill library, verification increases Recovery by up to 9.2%p and reduces or maintains Regression across all settings. The resulting library achieves the lowest Regression across all settings and the highest Recovery in all but one setting. It is the only library with higher Recovery than Regression in every setting, consistently improving SR over No Skill. Together, these results show that verification selects an effective subset of the original library.

Gains from verification are bounded by the reusable skills in the original library. On ALFWorld, Gemini and Qwen(27B) already solve most held-out tasks without skills, which leaves little room for verification to improve SR, yet SKILLSANDBOX is the only library that does. On WebShop, where the executors solve far fewer tasks on their own, the verified library adds more than 10%p with lower Regression. Verification selects from the original library rather than improving the skills themselves, so it can add only what that library contains. Indeed, far fewer of the ALFWorld skills are helpful on held-out tasks (Section 5.2).

5 ANALYSIS

To evaluate the verdicts, we compare each verdict against the skill's observed effect on held-out tasks. For each skill, we execute the held-out tasks with and without it (Appendix B) and label the skill helpful if task success increases and harmful if it decreases. The main experiments show that the verified library improves downstream performance. We now examine which components of our verification framework contribute to this improvement and whether it reflects accurate assessment of skill reusability through the following research questions:

- **RQ1:** *Does scenario synthesis improve the effectiveness of verification?*
- **RQ2:** *Does verification predict skill reusability?*
- **RQ3:** *Does reliable skill verification require a stronger model?*
- **RQ4:** *What do the synthesized scenarios reveal about skill reusability?*

5.1 RQ1: DOES SCENARIO SYNTHESIS IMPROVE THE EFFECTIVENESS OF VERIFICATION?

To evaluate the contribution of scenario synthesis, we compare verification evidence collected from synthesized scenarios, source tasks, and randomly selected tasks. We use Gemini 3.1 Flash-Lite on both benchmarks and hold the VERIFIER and rollout budget fixed, varying only the tasks on which evidence is collected. Under this budget, each skill is verified on five synthesized scenarios, five executions of its source task, or five random tasks.

Synthesized scenarios yield the highest verdict F1 and the best-performing skill library on both benchmarks (Figure 4), with held-out SR up to 15.7%p above the alternatives. On ALFWorld, the verdicts from source and random tasks barely separate helpful from harmful skills, and the libraries they retain fall below No Skill, with random-task verification even below the unverified library. Source tasks expose the skill-relevant situation only in its original context and random tasks rarely expose it at all, so neither provides the evidence verification requires.

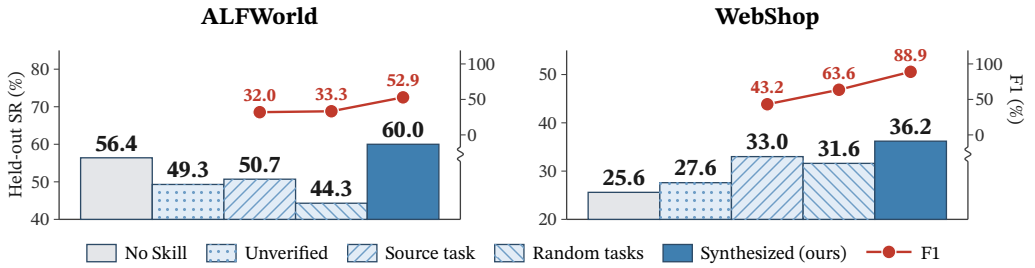


Figure 4: Verification with evidence from different types of tasks: held-out SR (bars, left axis) and F1 of the verdicts against held-out skill effects (line, right axis).

5.2 RQ2: Does verification predict skill reusability?

Table 3 reports precision, recall, F1, and specificity of the verdicts on both benchmarks. On WebShop, verification closely predicts the held-out utility of retained skills. High precision and specificity indicate that verification reliably retains helpful skills on held-out tasks while filtering out harmful ones. The 82.8% recall shows that verification preserves most helpful skills, although some are incorrectly rejected. Thus, on WebShop, a KEEP verdict provides a strong indication that the skill remians useful beyond its source experience.

Table 3: Verdict agreement with held-out skill effects.

PROPOSER	Verdict agreement (%)			
	Prec.	Rec.	F1	Spec.
WebShop				
Gemini 3.1 Flash	**96.0**	**82.8**	**88.9**	**94.4**
GPT-5.6 Luna	95.2	69.0	80.0	**94.4**
ALFWorld				
Gemini 3.1 Flash	**39.1**	**81.8**	**52.9**	**60.0**
GPT-5.6 Luna	34.8	72.7	47.1	57.1

Overall agreement is lower on ALFWorld, but verification still preserves the useful skills and improves the composition of the library. Only 23.9% of the ALFWorld skills are helpful on held-out tasks, which lowers precision. Verification nevertheless retains 81.8% of the helpful skills, nearly the same recall as on WebShop, and helpful skills make up 39.1% of the verified library, which is higher than the unverified library. Verification thus selects a more useful subset from execution evidence even where helpful skills are scarce.

5.3 RQ3: Does reliable skill verification require a stronger model?

Our framework targets self-evolving agents, which the quality of verification must therefore not depend on the capability of the model. To test this property, we replace only the PROPOSER with a stronger model (GPT-5.6 Luna (OpenAI, 2026)). The two PROPOSERs in Table 3 share the skills, the executor, the BUILDER, and the VERIFIER. Comparing their verdicts against the same held-out effects evaluates whether a stronger model is essential for reliable verification.

On both benchmarks, the executor's own model matches or exceeds GPT-5.6 Luna in every metric and retains more of the helpful skills. The stronger PROPOSER is not more accurate, since it rejects more helpful skills but no more harmful ones. Reliable verdicts therefore do not depend on the intuitive judgment that only a stronger model could provide. This result follows from the division of scenario synthesis into planning and construction, since the LLM is used only for planning and rollouts, and construction and scoring are fixed procedures. Structured execution evidence thus lowers the dependence of verification on model capability, and a self-evolving agent can select useful skills from its own execution evidence without an external model stronger than itself.

5.4 RQ4: What do the synthesized scenarios reveal about skill reusability?

Figure 5 analyze the skills' reusability according to three verification criterion executability, utility, and efficiency (Section 3.2) across all models. Executability is the number of scenarios with at least one action assigned $e = 1$, indicating that the action implements the skill's `principle`. Utility is the gain of the success rate between paired runs. Efficiency compares remaining action counts from the skill-used step to termination. In case of efficiency, we include only success pairs in both runs.

On WebShop, agents take more actions on average with skills, even when both with and without-skill runs succeed. The success gains of Qwen(27B) and Gemini therefore coexist with overhead

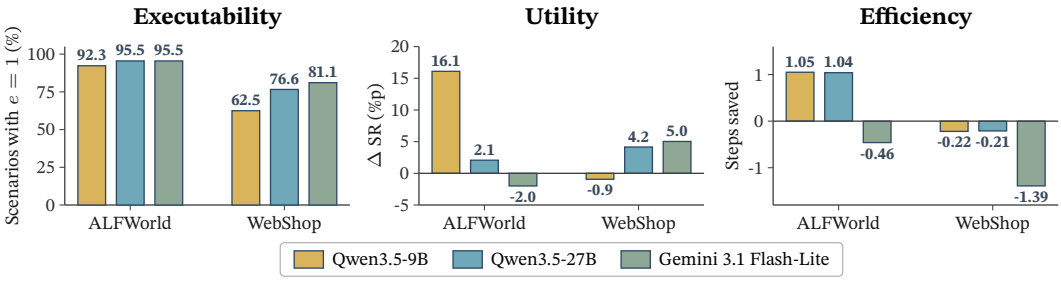


Figure 5: Executability, utility, and efficiency across all executors and benchmarks.

on already solvable tasks. This overhead cannot be explained solely by completing tasks that would otherwise fail. On ALFWorld, both Qwen models instead gain in success and save actions, showing that skill guidance can also yield successful execution. Gemini exhibits a different failure pattern there: despite high executability, skill lowers success and increases action counts, exposing a limitation in the benefit of the guidance beyond whether it can be followed.

6 Related Work

Skill acquisition and curation. Agents acquire reusable skills by abstracting past executions into programs or workflows that can be retrieved and applied in subsequent tasks (Wang et al., 2023; 2025b;a). A complementary approach distills successful and failed experiences into textual insights, reasoning strategies, or procedural instructions that guide future decisions (Zhao et al., 2024; Ouyang et al., 2026b; Fang et al., 2026b). Feedback-driven curation provides an implicit form of verification, using subsequent execution outcomes to revise stored guidance or learn policies for managing the skill repository (Fang et al., 2026b; Zhang et al., 2026; Ouyang et al., 2026a). More explicit verification evaluates the effects of individual skills through source-task tests or paired runs with and without the skill, informing refinement and retention decisions (Wang et al., 2025a; Ma et al., 2026). However, these methods can leave skills insufficiently verified, as the verification evidence they provide are neither skill-relevant nor new. We identify the lack of reliable opportunities to evaluate each skill as a central verification problem.

Task and environment synthesis. Recent approaches synthesize tasks and environments to train LLM agents. Task generation expands training opportunities within available environments (Zhou et al., 2025; Sun et al., 2025; Qi et al., 2025; Zhou et al., 2026). Environment synthesis constructs the interaction setting itself through executable environments or LLM-based simulators (Chae et al., 2026; Fang et al., 2026a; Song et al., 2026; Chen et al., 2026b). Adaptive approaches use the learner's performance to guide generation toward suitable difficulty levels or weaknesses in the current policy (Dennis et al., 2020; Zala et al., 2024; Guo et al., 2025). However, these objectives are insufficient for skill verification on their own, as they do not ensure that execution encounters the situation that the skill targets. To address the challenges in skill verification, we propose a framework that dynamically synthesizes tasks and environment for each target skill at verification time.

7 Conclusion

We introduced SkillSandbox to assess whether skills distilled from experience remain useful in new task contexts. The framework constructs a task and its environment around each skill, preserving its applicability condition while varying the context in which its guidance is exercised. Paired executions with and without the skill provide evidence of executability, utility, and efficiency for deciding whether to retain it. Across ALFWorld and WebShop with three models, verification improves downstream task success and execution efficiency while reducing harmful skill reuse. Our analyses show that synthesized scenarios provide more effective verification evidence than source replay or randomly selected tasks, and that reliable verdicts can be obtained using the executor's own model. These findings highlight the importance of constructing situations in which a skill can

be tested beyond its source experience, providing a practical basis for self-evolving agents to select reusable knowledge from their own experience.

AI USE STATEMENT

We used LLMs to assist with manuscript writing and polishing for clarity and readability, as well as with implementing code for our experiments. The authors reviewed all AI-assisted text and manually verified all AI-generated code before using it in the experiments. The authors take full responsibility for the final content of this work, including all AI-assisted text and code.

REFERENCES

Huan ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Qihan Ren, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence, 2026. URL https://arxiv.org/abs/2507.21046.

Anthropic. Agent Skills. GitHub repository, 2025. URL https://github.com/anthropics/skills. Accessed: 2026-09-26.

Hyungjoo Chae, Jungsoo Park, and Alan Ritter. Safe and scalable web agent learning via recreated websites. *arXiv preprint arXiv:2603.10505*, 2026.

Kunfeng Chen, Qihuang Zhong, Juhua Liu, and Bo Du. Skillcat: Contrastive assessment and topology-aware skill self-evolution for llm agents. *arXiv preprint arXiv:2606.13317*, 2026a.

Zhaorun Chen, Zhuokai Zhao, Kai Zhang, Bo Liu, Qi Qi, Yifan Wu, Tarun Kalluri, Xuefei Cao, Yuanhao Xiong, Haibo Tong, et al. Scaling agent learning via experience synthesis. In *International Conference on Learning Representations*, volume 2026, pp. 121394–121420, 2026b.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33:13049–13061, 2020.

Gen Dong, Yanjie Gao, Liqun Li, Tianyin Xu, Yu Hua, and Fan Yang. Agent skills can be harmful: An empirical study of skill-induced failures in llm agents. *arXiv preprint arXiv:2608.11888*, 2026.

Runnan Fang, Shihao Cai, Baixuan Li, Jialong Wu, Guangyu Li, Wenbiao Yin, Xinyu Wang, Xiaobin Wang, Liangcai Su, Zhen Zhang, et al. Towards general agentic intelligence via environment scaling. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17610–17621, 2026a.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17490–17502, 2026b.

Srishti Gautam, Arjun Radhakrishna, and Sumit Gulwani. Skillaxe: Sharpening llm-authored agent skills through evaluation-guided self-refinement. *arXiv preprint arXiv:2606.10546*, 2026.

Google DeepMind. Gemini 3.1 Flash-Lite. Model card, March 2026. URL https://deepmind.google/models/model-cards/gemini-3-1-flash-lite/. Accessed: 2026-09-26.

Jiacheng Guo, Ling Yang, Peter Chen, Qixin Xiao, Yinjie Wang, Xinzhe Juan, Jiahao Qiu, Ke Shen, and Mengdi Wang. Genenv: Difficulty-aligned co-evolution between llm agents and environment simulators. *arXiv preprint arXiv:2512.19682*, 2025.

Tingxu Han, Yi Zhang, Wei Song, Chunrong Fang, Zhenyu Chen, Youcheng Sun, and Lijie Hu. Swe-skills-bench: Do agent skills actually help in real-world software engineering? *arXiv preprint arXiv:2603.15401*, 2026.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101:99–134, 1998. URL https://api.semanticscholar.org/CorpusID:5613003.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, et al. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026.

Sirui Liang, Pengfei Cao, Jian Zhao, Wenhao Teng, Xiangwen Liao, Jun Zhao, and Kang Liu. Learning how to remember: A meta-cognitive management method for structured and transferable agent memory. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 30733–30753, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1535. URL https://aclanthology.org/2026.findings-acl.1535/.

Yuxuan Liu, Zhaochen Su, Lingyun Xie, Yuhao Zhang, Qing Zong, Jiahe Guo, Zhongwei Xie, Yiyan Ji, Yauwai Yim, Hongyu Luo, et al. Skillrevise: Improving llm-authored agent skills via trace-conditioned skill revision. *arXiv preprint arXiv:2606.01139*, 2026.

Yuchen Ma, Yue Huang, Han Bao, Haomin Zhuang, Swadheen Shukla, Michel Galley, Xiangliang Zhang, and Stefan Feuerriegel. Skillgen: Verified inference-time agent skill synthesis. *arXiv preprint arXiv:2605.10999*, 2026.

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

OpenAI. GPT-5.6, 2026. URL https://deploymentsafety.openai.com/gpt-5-6/gpt-5-6.pdf.

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

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, volume 2026, pp. 94327–94354, 2026b.

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

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

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

Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Envscaler: Scaling tool-interactive environments for llm agent via programmatic synthesis. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 8326–8357, 2026.

Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, et al. Os-genesis: Automating gui agent trajectory construction via reverse task synthesis. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5555–5579, 2025.

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

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks. *arXiv preprint arXiv:2504.06821*, 2025a.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 2025b. URL https://proceedings.mlr.press/v267/wang25bx.html.

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

Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Zirui Liu, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. How memory management impacts LLM agents: An empirical study of experience-following behavior. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 623–645, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.27. URL https://aclanthology.org/2026.acl-long.27/.

Cheng Yang, Xuemeng Yang, Licheng Wen, Daocheng Fu, Jianbiao Mei, Rong Wu, Pinlong Cai, Yufan Shen, Nianchen Deng, Jia Xu, Botian Shi, Yu Qiao, and Haifeng Li. Towards self-evolving agents: Enabling autonomy through interactive experience refinement. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 30424–30451, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1522. URL https://aclanthology.org/2026.findings-acl.1522/.

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

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

Abhay Zala, Jaemin Cho, Han Lin, Jaehong Yoon, and Mohit Bansal. Envgen: Generating and adapting environments via llms for training embodied agents. *arXiv preprint arXiv:2403.12014*, 2024.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, volume 2026, pp. 86069–86100, 2026.

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

Yifei Zhou, Qianlan Yang, Kaixiang Lin, Min Bai, Xiong Zhou, Yu-Xiong Wang, Sergey Levine, and Li Erran Li. Proposer-agent-evaluator (pae): Autonomous skill discovery for foundation model internet agents. In *Forty-second International Conference on Machine Learning*, 2025.

Yifei Zhou, Sergey Levine, Jason Weston, Xian Li, and Sainbayar Sukhbaatar. Self-challenging language model agents. *Advances in Neural Information Processing Systems*, 38:113959–113991, 2026.

APPENDIX CONTENTS

A Baseline Details **14**

B Implementation Details **14**

C Scenario Synthesis **15**

D Case Study **16**

E Additional Results **17**

F Prompts **19**

F.1 Executor 19

F.2 Skill distillation 20

F.3 PROPOSER 20

A BASELINE DETAILS

We distinguish *no verification* (unconditionally retaining generated skills) from *execution-based verification*, which includes feedback-driven skill admission, revision, removal, or reweighting. The latter uses either *source replay* (repeated attempts at the original task) or *downstream tasks* (subsequent source tasks selected independently of a skill). All libraries are built from the same 100 source tasks and frozen before held-out evaluation.

Vanilla Skill. *No verification.* Each source trajectory is distilled into an applicability condition and an actionable principle using the skill distillation prompts in Appendix F.2. All generated skills are retained; SKILLSANDBOX verifies skills from this same initial pool.

ReasoningBank. *No verification.* ReasoningBank sequentially solves source tasks using prior memories, judges each outcome, and extracts up to three structured memories from each trajectory. Our basic variant, without MaTTS, appends all extracted items without revising or deleting existing ones: the judge assesses task success, not memory utility.

MemP. *No verification.* We use MemP's *Script* representation with *Vanilla* updates. The executor sequentially solves source tasks using retrieved scripts; after each task, a builder appends one workflow summary regardless of success, without utility-based filtering, revision, or deletion.

ExpeL. *Execution-based verification: source replay.* ExpeL collects source trajectories before extracting insights. After failure, our executor reflects and retries the same task at most once. Using the same model, the extractor compares successful and failed attempts and groups of successes, curating rules through `ADD`, `EDIT`, `AGREE`, and `REMOVE` with importance counters.

ACE. *Execution-based verification: downstream tasks.* ACE solves tasks with an evolving playbook, reflects on entry helpfulness, and incrementally curates the playbook. Our implementation makes three source passes, adding entries and updating feedback counters, with separate duplicate merging and budget-triggered pruning of harmful entries. Held-out evaluation uses the entire frozen playbook rather than top-three retrieval.

SkillOS. *Execution-based verification: downstream tasks.* We use training-free *SkillOS-base* (**?**), with the paper's Appendix A prompts. The executor makes one ordered source pass, retrieving skills from the evolving repository with BM25. After each task, the curator uses its trajectory, outcome, and retrieved skills to insert, update, or delete skills before the next task.

B IMPLEMENTATION DETAILS

Executor. Table 4 summarizes the executor settings for the two benchmarks. At each step, the executor receives the current observation and admissible actions together with the recent interaction history. The executor prompts are provided in Appendix F.1.

Table 4: Executor settings. History length counts preceding observation–action pairs; the current observation is supplied separately.

Setting	ALFWorld	WebShop
Action history length	10	5
History content	Observation–action pairs	Observation–action pairs
Maximum steps per episode	30	15

Skill distillation and retrieval. We distill each source trajectory into an applicability condition and an actionable principle (Appendix F.2). On both benchmarks, held-out retrieval ranks full skill texts against the task query by cosine similarity using `text-embedding-3-small`, supplying the top three skills in descending order.

For Question I (Section 2), GPT-5.6 Luna extracts fixed conditions from `When to apply`, consulting `Principle` for context, then checks them against the task objective and with-skill trajectory. OBSERVED requires evidence for all conditions and their logical and temporal relationships; state and history conditions require trajectory evidence, not task outcomes or claimed skill use. Otherwise, the label is NOT OBSERVED. We report the mean number of OBSERVED tasks per skill across 50 skills, each evaluated on 10 retrieved tasks.

For Question II, Gemini 3.1 Flash-Lite executes a fixed random stream of 500 held-out WebShop tasks using the same 50-skill library, with three repeats per setting: Full Library, Top-3, and Top-5 Retrieval. Retrieval uses the embedding method above. The ReAct executor cites skill identifiers when using them; at each stream prefix, we report the percentage of skills cited in at least one or five distinct tasks.

VERIFIER. The VERIFIER compares trajectories τ^+ and τ^- produced by the same executor in the same synthesized scenario, with and without skill k, respectively. We align the full trajectories by maximizing the number of pairs with identical pre-action observations, preserving temporal order and using each step at most once. Ties are resolved by choosing the lexicographically earliest sequence of paired step indices, with the τ^+ index first. For each distinct observation, we retain only the earliest aligned pair whose step in τ^+ is marked as using skill k. From the same observation, we examine how using the skill changes the agent's subsequent behavior compared with execution without the skill.

For each retained pair, let $t^{\pm}$ denote the matched step indices and $T^{\pm}$ the termination indices of the corresponding trajectories. The remaining action counts $T^{\pm} - t^{\pm}$ include all subsequent actions in both trajectories, including those taken when the same observation recurs. To assess utility, we compare terminal rewards $r^{\pm} \in \{0, 1\}$, which are 1 for task success and 0 for failure under the scenario's success criterion. To account for efficiency, we discount these rewards back to the matched steps so that successful continuations requiring fewer actions receive higher returns:

$$\Delta u = r^{+}\gamma^{T^{+}-t^{+}} - r^{-}\gamma^{T^{-}-t^{-}}, \qquad 0 < \gamma < 1. \tag{2}$$

Executability concerns whether the `principle` can be executed as admissible actions supported by the environment. The VERIFIER uses agreement between the `principle` and an action in τ^+ as evidence of this feasibility at each matched point. It sets $e = 1$ when the action implements the corresponding instruction and $e = 0$ when this evidence is absent. Only points with observed evidence of executability contribute to $R(k)$. We first average $e\,\Delta u$ over the retained pairs within each trajectory pair, then across synthesized scenarios. This empirical averaging gives the reusability score and KEEP/REJECT verdict in Equation 1.

Evaluation protocol. For RQ1 (Section 5.2), we execute the 500 held-out WebShop tasks with and without each of the 50 evaluated skills. Let Help_k denote the number of held-out tasks that skill k turns from failure to success and Harm_k the number it turns from success to failure; the net held-out effect of the skill is $E_k = (\mathrm{Help}_k - \mathrm{Harm}_k)/500$. Spearman's ρ, with average ranks assigned to ties, and Pearson's r are computed between E_k and the reusability score $R(k)$ assigned by the verification pipeline over all 50 skills, including the three with zero net effect. For the agreement metrics, a skill is labeled helpful when $\mathrm{Help}_k > \mathrm{Harm}_k$ and harmful when $\mathrm{Help}_k < \mathrm{Harm}_k$.

C SCENARIO SYNTHESIS

Synthesis Types The synthesis typology of Section 5.4 covers scenarios generated for WebShop and ALFWorld skills in the Qwen3.5-27B setting. For each environment, GPT-5.6 Luna first summarizes each scenario's operations and implementation plan in one or two sentences and induces types from these summaries. To categorize how skills cause harm, we select the scenarios, in which the executor fails with the skill, succeeds without it, and receives a negative scenario score. Given the skill, both trajectories, and their outcomes, the same model explains how the skill could have contributed to the failure, and the explanations are grouped into types in the same way. Figure 6 reports the share of scenarios in each synthesis type; types are not exclusive, so shares sum to more than 100%.

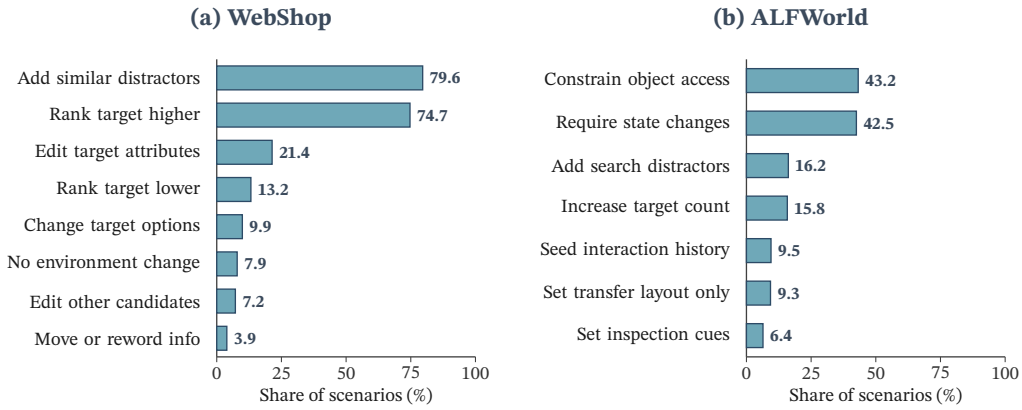


Figure 6: Planned synthesis mechanisms for Qwen3.5-27B on (a) WebShop and (b) ALFWorld. Each bar shows the percentage of scenarios using that transformation. A scenario may involve multiple transformations.

D CASE STUDY

Table 5: Case study of rejected skills. The shaded block is the skill under verification, and the red bold text marks the part of its `principle` that causes the rejection. τ^+ and τ^- denote the executions of the same synthesized scenario with and without the skill, and the highlighted action is the one guided by the skill.

ALFWorld Verdict: REJECT

Task *Examine the tissuebox with the desklamp*

`condition`: The task requires using a stationary appliance on a movable object located in a different area.
`principle`: **Locate and activate the stationary tool first**, retrieve the target and transport it to the tool's location.

Synthesized Scenario	**Execution Result**	
Task *put a cool bread in countertop* **Environment** • bread on the diningtable, **not cooled** • fridge in a **different area**, closed • agent starts at the diningtable; the bread can be picked up immediately	τ^+**: Fail, 30 steps** 1 `go to fridge` 2 `open fridge` 3 `close fridge` 4 `go to cabinet` · · · searches other receptacles; never returns for the bread	τ^-**: Success, 6 steps** 1 `take bread from diningtable` 2 `go to fridge` 3 `open fridge` 4 `cool bread with fridge` ✓ 5 `go to countertop` 6 `put bread on countertop` ✓

WebShop Verdict: REJECT

Task *I need some hair drying towels that are easy to clean, and price lower than 50.00 dollars*

`condition`: Evaluate a product for an easy-to-clean requirement after confirming its price and category criteria.
`principle`: Verify the claim in **both Features and Description** before purchase.

Synthesized Scenario	**Execution Result**	
Task *I want to purchase a hairdressing backpack with an easy clean feature for under 50.* **Environment** • required attribute `easy clean`, price at most $50, product kind hairdressing backpack • search query `hairdressing backpacks easy to clean`; the episode starts on the results page • target product placed at **rank 3**	τ^+**: Fail, 15 steps** · · · 4 `click[Description]` 5 `click[Back to Search]` 6 `search[same query]` · · · 7–15: reopens the product, alternates the two sections; no purchase	τ^-**: Success, 4 steps** · · · 4 `click[Buy Now]` ✓

Table 5 illustrates how synthesized scenarios expose the skill-relevant situation and use it as verification evidence.

ALFWorld. In the source task, the agent successfully examines a tissue box with a desk lamp by activating the lamp before retrieving the tissue box from another location. This ordering works because the lamp can be activated independently and remains on while the agent retrieves the object. The distilled skill generalizes this source-specific ordering into a broader principle: activate a stationary appliance before retrieving the target object. To reproduce the skill's condition, the synthesized scenario places a stationary appliance (fridge) and a movable object (bread) at different locations. The Builder places the uncool bread at the agent's starting location and keeps the fridge closed elsewhere, requiring the agent to cool the bread before placing it on a countertop. Unlike the lamp, cooling requires the agent to hold the bread at the fridge. Nevertheless, τ^+ follows the skill and goes to the fridge first, then fails to return for the bread and reaches the step limit. In contrast, τ^- takes the bread first, carries it to the fridge, cools it, and completes the task in six steps. The scenario thus shows that the principle distilled from the source does not transfer to this new context, leading the verifier to reject the skill.

WebShop. The source task asks for an easy-to-clean hair-drying towel under $50. During the source experience, the agent finds a candidate whose cleaning information appears in both the Features and Description sections, motivating a skill that requires checking both before purchase. For verification, the synthesized task instead asks for an easy-to-clean hairdressing backpack under $50 and places a qualifying product at rank 3 of the search results. This creates a new instance of the same product-verification situation, but one in which sufficient evidence is already available from a single section. After inspecting the Features, τ^- treats the requirement as satisfied and purchases the product in four steps. τ^+, however, follows the principle and continues to seek confirmation from the Description. Because the same claim is absent there, it repeatedly revisits the product sections and exhausts the action budget without purchasing. The synthesized scenario therefore exposes that requiring evidence from both sections is not generally necessary, but reflects the information layout encountered in the source task, leading the verifier to reject the skill.

E Additional Results

WebShop score correlation. Table 6 supplements RQ1 with the correlations between the verification score $R(k)$ and the net held-out effect E_k on WebShop, using the evaluation protocol in Appendix B. Both PROPOSERs have positive Spearman correlations, indicating that skills with higher verification scores tend to have larger held-out effects. Pearson's r provides a supplementary check of a linear relation; it does not establish that the score is calibrated to the magnitude of the improvement.

Table 6: WebShop score correlation with held-out skill effects. ρ and r denote Spearman and Pearson correlations.

PROPOSER	Spearman ρ	Pearson r
Gemini 3.1 Flash	**0.665**	0.552
GPT-5.6 Luna	0.662	**0.645**

Retrieval size. Table 7 examines how the number of retrieved skills affects downstream task success. We vary the retrieval size $k \in \{1, 3, 5\}$ with Gemini 3.1 Flash-Lite as the executor, while keeping the KEEP/REJECT verdicts, the retained skill pool, the held-out tasks, and the executor settings fixed. For each task, we rank the retained skills by cosine similarity between embeddings of the task instruction and the full skill text, using the same embedding model as in the main experiments. The top-k skills are injected into the executor's prompt in descending order of similarity. All settings share the same ranking, so the top-1 and top-3 selections are prefixes of the top-5 selection. The $k = 3$ column coincides with the Gemini 3.1 Flash-Lite SKILLSANDBOX row of Table 2. We set $k = 3$ in all other experiments because it achieves the best SR on both benchmarks, tying with $k = 5$ on ALFWorld and outperforming both $k = 1$ and $k = 5$ on WebShop.

Causes of rejection. Figure 7 shows, for each executor and verification criterion, the primary cause of failure among the rejected skills. A rejected skill can fail several criteria; within each

Table 7: Held-out SR (%) of Gemini 3.1 Flash-Lite under different retrieval sizes k, with the number of solved tasks in parentheses. $k = 3$ is the setting of all other experiments. Bold marks the best value in each row.

Benchmark	$k = 1$	$k = 3$	$k = 5$
ALFWorld (140 tasks)	58.6 (82)	**60.0** (84)	**60.0** (84)
WebShop (500 tasks)	31.4 (157)	**36.2** (181)	35.0 (175)

criterion, one primary cause is assigned per skill, and the three largest cause types are shown with the rest pooled as Other.

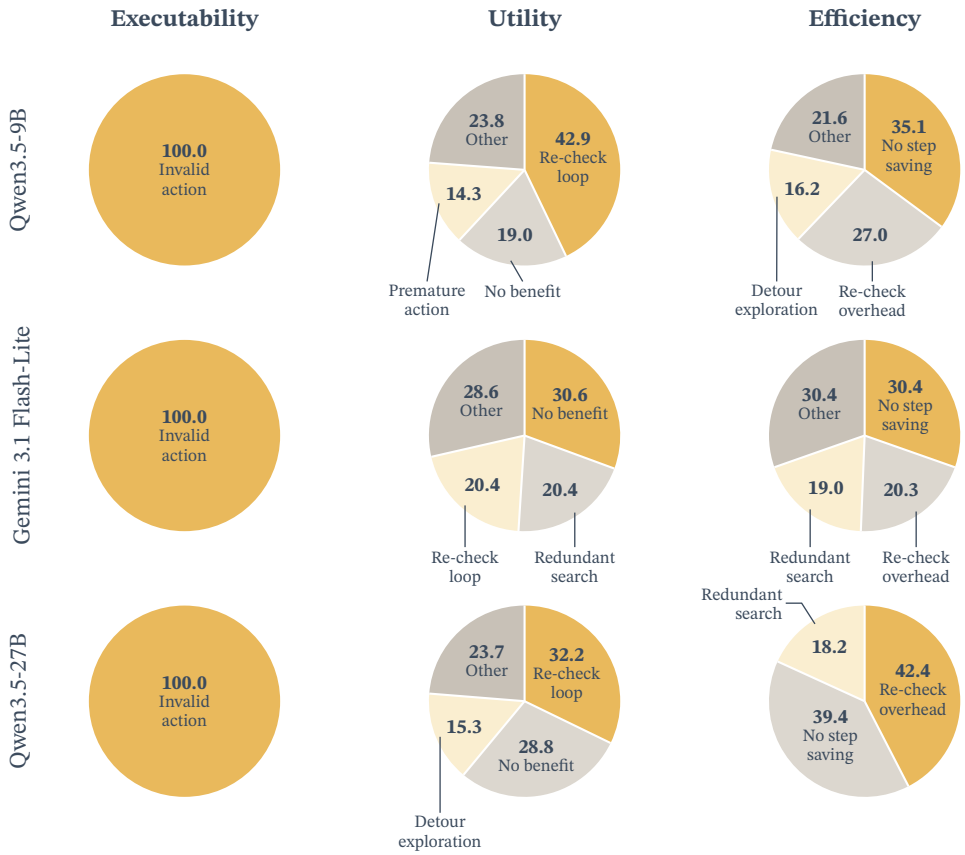


Figure 7: Primary cause of failing each verification criterion among the rejected skills of each executor, as the share of the skills that failed the criterion. Causes are typed by GPT-5.6 Luna from the verification trajectories, one primary cause per skill and criterion; the three largest types of each criterion are shown and the rest are pooled as Other.

F PROMPTS

We present prompt templates for execution and skill distillation, and structured summaries of the scenario-proposal prompts. Braced placeholders denote runtime inputs, such as a task, trajectory, skill, or environment interface. The proposal summaries describe the inputs and output schemas for each stage; retry and format-repair messages are omitted.

F.1 EXECUTOR

The following prompts execute tasks without skills, using the current task, observation, admissible actions, and recent action history.

Executor: ALFWorld

Follow the user's ALFRED task instructions.
Return only JSON with fields in this order: {`"rationale": "step-by-step reasoning", "action": "one exact admissible action"`}.

You are an expert agent operating in the ALFRED Embodied Environment. Your task is to: {`task`}
Below are the most recent {`history_length`} observations and the corresponding actions you took: {`action_history`}
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. Once you've finished your reasoning, choose one admissible action for the current step.
Return only JSON: {`"rationale": "step-by-step reasoning", "action": "one exact admissible action"`}.

Executor: WebShop

You are an expert autonomous agent operating in the WebShop e-commerce environment.
Your task is to: {`task_description`}.
Below are your most recent {`history_length`} interaction steps (observations when provided, and actions):
{`action_history`}

You are now at step {`current_step`} out of {`total_steps`} steps, and your current observation is: {`current_observation`}

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

Now it's your turn to take one action for the current step. You should first reason step-by-step about the current situation, then think carefully which admissible action best advances the shopping goal.

- Thought: Reasoning process
- Action: Return exactly one executable admissible command for current step

Return only a valid JSON object with exactly these two fields, using `rationale` for Thought and `action` for Action: {`"rationale": "the complete Thought described above", "action": "the single executable Action described above"`}

F.2 SKILL DISTILLATION

These prompts generate one skill from the source trajectory, before verification or revision. The trajectory input includes the source task, observed transitions, recorded actions, and execution outcome. The output consists of a `When to apply` condition and a `Principle` in one or two sentences. We distinguish prompts for successful and failed source trajectories.

Skill distillation: ALFWorld failure

You are an expert at extracting reusable skills from agent failures. Analyze the failed trajectory from an AI agent operating in the ALFRED embodied household environment (ALFWorld).

Trajectory: {`trajectory`}

Generate one concise skill that captures the reusable lesson from this failure.

- When to apply: Specific trigger condition.
- Principle: The core actionable insight in 1-2 sentences.

Skill distillation: ALFWorld success

You are an expert at extracting reusable skills from agent successes. Analyze the successful trajectory from an AI agent operating in the ALFRED embodied household environment (ALFWorld).

Trajectory: {`trajectory`}

Generate one concise skill that captures the reusable lesson from this success.

- When to apply: Specific trigger condition.
- Principle: The core actionable insight in 1-2 sentences.

Skill distillation: WebShop failure

You are an expert at extracting reusable skills from agent failures. Analyze the failed trajectory from an AI agent operating in an online shopping environment (WebShop).

Trajectory: {`Trajectory`}

Generate one concise skill that captures the reusable lesson from this failure.

- When to apply: Specific trigger condition.
- Principle: The core actionable insight in 1-2 sentences.

Skill distillation: WebShop success

You are an expert at extracting reusable skills from agent successes. Analyze the successful trajectory from an AI agent operating in an online shopping environment (WebShop).

Trajectory: {`Trajectory`}

Generate one concise skill that captures the reusable lesson from this success.

- When to apply: Specific trigger condition.
- Principle: The core actionable insight in 1-2 sentences.

F.3 PROPOSER

For brevity, we summarize the proposal prompts by their inputs, core instructions, and required JSON outputs. Each summary combines condition interpretation and scenario specification for one environment.

Proposer: ALFWorld

Input

<skill_condition>
<source_task>
<source_trajectory_summary>
<environment_interface>
<skill>
<interpreted_requirements>
<compatible_or_fixed_goals>
<earlier_scenario_summaries>
<selected_test_specification> (optional)

Core instructions

1. Preserve the full condition, including alternatives, negation, quantities, and prior events. Separate task objectives from the current state and execution history; do not derive applicability requirements from the principle.

2. Determine whether the condition is established by the initial state, real setup history, or the executor's own rollout. Leave exploration and manipulation to the executor when the condition arises during execution.

3. Specify symbolic roles and supported construction operations. Vary objects, placements, and compatible task context while preserving required capabilities and any fixed goal; concrete entities are bound later.

4. Keep the goal and the skill-guided decision unfinished. Do not invent observations or capabilities, pre-execute the principle, or design the scenario to guarantee a favorable verdict.

Output

Condition interpretation: A JSON object with `status`, `encounter_mode`, `encounter_reason`, `history_required`, and `task_checks`.

Scenario specification: One symbolic program with `roles`, `operations`, `prepare_start`, `setup_events`, `condition_checks`, `test_checks`, and `novelty`; include `goals` only when no fixed goal is supplied. Condition checks concern the encounter, while test checks concern the initial configuration.

Proposer: WebShop

Input

<skill_condition>
<environment_interface>
<skill>
<interpreted_requirements>
<fixed_target_product>
<linked_human_task>
<scoring_conditions>
<earlier_scenario_summaries>
<response_schema>
<frozen_design>
<operator_definitions>

Core instructions

1. Restate every condition clause without adding restrictions from the principle. Preserve alternatives, history, and local absence claims; report a missing native capability rather than weaken the condition.

Proposer: WebShop (continued)

2. Keep the fixed target's identity and product type. Preserve the linked task and scoring conditions except for changes needed to reproduce the condition; use existing product facts whenever sufficient.

3. Specify minimal supported edits to product records, options, or result lists. Start at the native search screen with no preselected options, and use one search query derived from the final task. Leave inspections and selections to the executor.

4. Translate the frozen design into symbolic operations and identify which requirements they establish. Do not fabricate observed evidence, pre-execute the principle, or guarantee that the skill helps; an empty operation list is valid when no edits are needed.

Output

Condition interpretation: A JSON object containing `status`, scoped `requirements` (task, catalog, or trajectory), and `rationale`; if unsupported, return `status` and `reason`.

Scenario design: A JSON object with `mode`, `intervention`, `goal`, `wording`, and `novelty`. Goal fields express requirements for subsequent binding.

Operation specification: A JSON object with `value_declarations`, `operations`, and `coverage`, linking requirements to edits and the intended evidence locations.