SKILLADM: RUBRIC-GUIDED, SKILL-SPECIFIC ADMISSION FOR ONLINE TEST-TIME SKILL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Online test-time skill learning (TTSL) enables frozen LLM agents to acquire reusable skills through interaction, continuously reusing and updating them over sequential task streams. Since a skill distilled from limited experience is not necessarily beneficial, skill admission—determining whether a candidate skill improves task execution before activating it for reuse—is critical for reliable online TTSL. Existing admission methods face two key challenges: coarse, oracle-free evaluation and validation limited to existing tasks. To address these, we propose SkillAdm, an online TTSL framework for continual agent self-improvement that acquires verified environment knowledge, constructs task-specific, progress-aware rubrics for oracle-free skill distillation, and performs two-stage utility assessment over both skill-specific validation tasks and relevant future verification tasks. Experiments across three benchmarks and three LLM backbones show that SkillAdm consistently outperforms the oracle-free baseline ASI and even surpasses oracle-assisted baselines using ground-truth outcomes.

1 INTRODUCTION

Large language models (LLMs) have enabled autonomous agents to reason, plan, and interact with complex environments. Yet agents often encounter tasks requiring knowledge or procedural skills beyond what their underlying models can readily provide. This calls for agents that learn from their own interactions and improve on subsequent tasks. Updating model parameters through supervised fine-tuning, preference optimization, or reinforcement learning on interaction trajectories (Zeng et al., 2024; Song et al., 2024; Qi et al., 2025) is one route, but can be computationally costly or infeasible during deployment, particularly when model weights are inaccessible.

An alternative is test-time skill learning (TTSL), which distills reusable skills through test-time interaction with the environment while keeping the LLM frozen. Beyond offline settings, where skills can be consolidated after collecting a batch of tasks, online TTSL targets sequential task streams in which newly acquired skills are continuously reused and updated. This setting better reflects deployment scenarios in which agents must improve on the fly. Existing online TTSL methods(Wang et al., 2025b; Xu et al., 2026; Zhang et al., 2026c) typically abstract execution trajectories into procedures, behavioral guidelines, or executable strategies and organize them into a skill memory that can be updated using new trajectories from later tasks.

However, a newly acquired skill is not necessarily beneficial. Skills distilled from limited experience may contain unnecessary actions or instructions that work only for the original task. When reused, an unreliable skill can cause mistakes on later tasks and affect the trajectories used to learn further skills. This highlights the need for skill admission in TTSL: determining whether a candidate skill improves task execution before activating it for reuse. Existing skill admission methods for TTSL primarily assess utility through trajectories generated with candidate skills. Although effective, these approaches face two key challenges.

L1: Coarse-Grained Oracle-Free Evaluation. As shown in Figure 1 (a), existing skill admission methods (Shang & Yang, 2026; Moll et al., 2026; Zhang et al., 2026a) mainly rely on ground-truth outcomes to evaluate trajectories and assess skill utility. However, such outcome supervision may be unavailable during deployment. To address this, ASI (Wang et al., 2025a), to our knowledge the only oracle-free method, directly uses an LLM to classify trajectories as successful or unsuccessful. Yet without an explicit rubric, LLM-based judgments provide no fine-grained criteria for evaluation,

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.

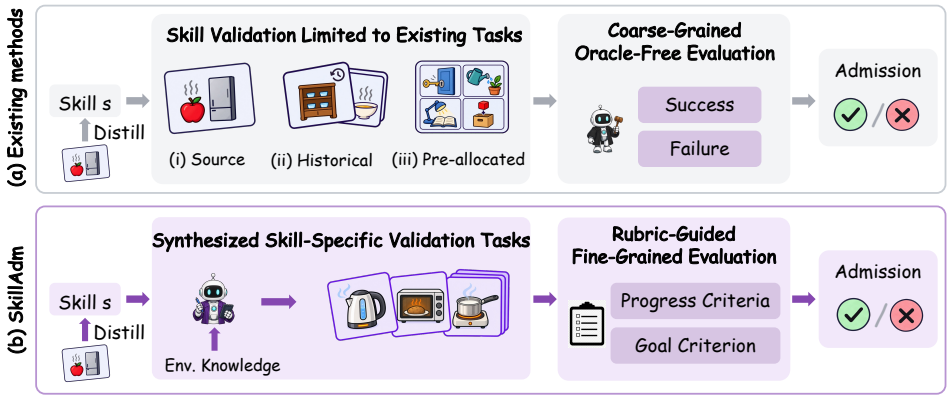

Figure 1: Comparison between existing skill admission methods (a) and SKILLADM (b). 1) Existing methods validate skills mainly on pre-existing tasks, whereas SKILLADM constructs skill-specific validation tasks to directly evaluate the target skill. 2) Existing methods mainly rely on ground-truth outcomes, with the only oracle-free method using LLM correctness judgments, whereas SKILLADM introduces task-specific rubrics for fine-grained, oracle-free evaluation.

limiting their ability to distinguish different levels of task progress and identify reusable behaviors. To overcome this limitation, we introduce task-specific rubrics that provide fine-grained criteria for evaluating skill utility, as shown in Figure 1 (b).

L2: Skill Validation Limited to Existing Tasks. As shown in Figure 1 (a), existing methods mainly validate skills on source tasks (Wang et al., 2025a), historical tasks (Moll et al., 2026; Yang et al., 2026; Qiu et al., 2026), or pre-allocated tasks (Shang & Yang, 2026; Shang et al., 2026). Although these methods can select relevant tasks, such tasks are not specifically designed to validate the skill under consideration. As a result, they may fail to reveal whether the skill is actually useful and reusable, since the validation signal is shaped by the original purpose of those tasks rather than by the skill itself. To overcome this limitation, as shown in Figure 1 (b), we construct skill-specific validation tasks to validate each skill.

To address these challenges, we propose SKILLADM, an online TTSL framework for continual agent self-improvement with the LLM kept frozen. It enables agents to autonomously discover, validate, and reuse skills without ground-truth outcomes. As illustrated in Figure 2, SkillAdm comprises four components. **Pre-Task Environment Knowledge Acquisition** first actively explores the environment to identify and verify reusable state–action–outcome relationships, providing environment knowledge that supports the remaining components. To address the first challenge, **Rubric-Guided Quality-Aware Skill Distillation** constructs task-specific, progress-aware rubrics from the task specification and acquired environment knowledge, then distills Action and Avoidance skills via contrastive trajectory analysis. To address the second challenge, **Skill-Specific Utility Validation** performs task-level, rubric-guided validation on newly constructed tasks tailored to each skill using the acquired environment knowledge. Finally, **Decoupled Online Utility Verification** examines whether the observed utility transfers to relevant future tasks, following existing practice. Only skills that meet the utility-margin and significance criteria in both stages are activated for retrieval. These skills then guide subsequent interactions, providing new experience for further skill learning.

Our contributions are threefold. (1) We develop task-specific, progress-aware rubrics grounded in verified environment knowledge to guide oracle-free skill distillation and utility assessment. (2) We introduce a two-stage skill admission mechanism that combines skill-specific validation tasks with independent verification on relevant future tasks. (3) Extensive experiments demonstrate that SkillAdm consistently outperforms the oracle-free baseline ASI across three benchmarks and three LLM backbones, and even surpasses oracle-assisted baselines that use ground-truth outcomes.

2 SKILLADM

In this section, we first formalize the problem, and then describe the four key components of SkillAdm.

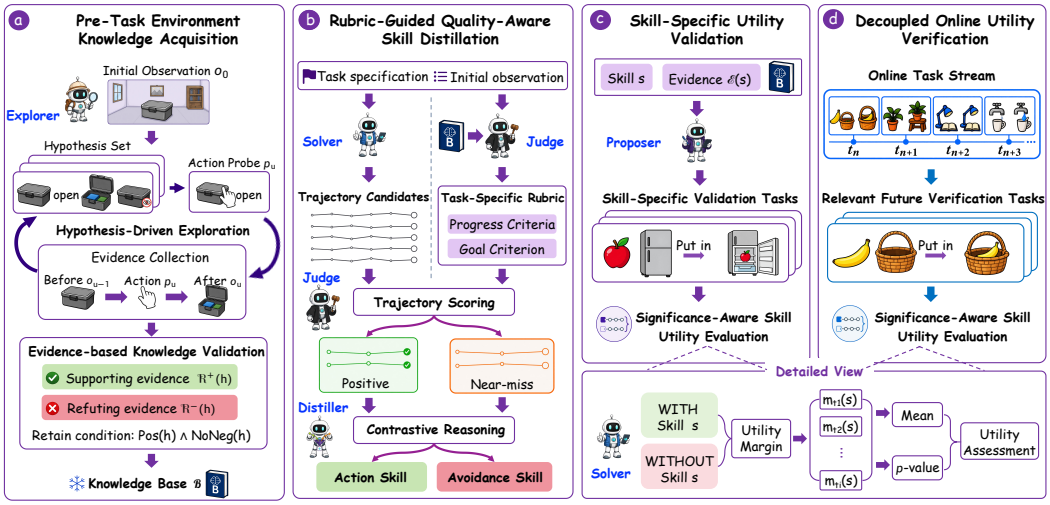

Figure 2: Overview of SkillAdm. (a) Acquires verified environment knowledge. (b) Distills Action and Avoidance skills with task-specific, progress-aware rubrics. (c) Assesses skill utility on constructed skill-specific tasks. (d) Verifies utility transfer on relevant future tasks.

2.1 Problem Formulation

We study oracle-free online skill admission for test-time skill learning without additional training. The LLM remains frozen, and neither skill discovery nor admission has access to ground-truth outcomes. The agent processes an online task stream $\mathcal{D} = \{t_1, t_2, \ldots, t_N\}$ in an interactive environment $\mathcal{W}$, where each task t_i specifies its objective and execution context. At round i, task t_i arrives, and the agent interacts with $\mathcal{W}$ to produce a trajectory from which new skills may be distilled. These skills are not immediately available for retrieval; instead, online admission determines whether each skill has sufficient utility for activation.

2.2 Pre-Task Environment Knowledge Acquisition

Inspired by pre-task environment exploration for agent planning and adaptation (Zhou et al., 2025; Chen et al., 2026a; Ye et al., 2026; Zhang et al., 2026b), we acquire reusable knowledge of relationships among environment states, actions, and observable outcomes to support subsequent task-specific rubric construction and executable skill-specific validation task generation. Towards this end, we develop an iterative, hypothesis-driven exploration strategy.

Hypothesis-Driven Exploration. Exploration starts from the initial observation o_0 returned by the environment, with an empty hypothesis set $\mathcal{H}_0$ and evidence buffer $\mathcal{R}_0$. Each hypothesis is defined as $h = (o_h, p_h, \hat{o}_h, o_h^-)$, where o_h denotes an abstract observation context for action p_h, $\hat{o}_h$ the expected outcome after executing p_h, and o_h^- a potential contradictory outcome used only for hypothesis validation. The abstract context helps avoid instance-specific hypotheses and promotes reusable action–outcome knowledge. For example, a hypothesis can be instantiated as $h =$ (“the container is closed”, “open the container”, “the contents become visible”, “the contents remain hidden”).

As shown in Figure 2, at iteration u, the Explorer first identifies an action–outcome hypothesis that is relevant to the current observation and accumulated evidence. Specifically, it retrieves a previously proposed hypothesis from $\mathcal{H}_{u-1}$ when it remains applicable to the current context, or generates a new hypothesis when existing hypotheses do not adequately explain the current evidence. This yields a single hypothesis h_u, based on which the Explorer then proposes an action probe p_u:

$$h_u = \text{ProposeHypothesis}(o_{u-1}, \mathcal{H}_{u-1}, \mathcal{R}_{u-1}),\ p_u = \text{ProposeProbe}(o_{u-1}, h_u, \mathcal{R}_{u-1}). \quad (1)$$

A probe may either directly test the hypothesis or serve as a prerequisite for a subsequent direct test. Executing p_u under observation o_{u-1} produces the resulting observation o_u. We record the transition as $r_u = (o_{u-1}, p_u, o_u)$, update the evidence buffer as $\mathcal{R}_u = \mathcal{R}_{u-1} \cup \{r_u\}$ and the hypothesis set as $\mathcal{H}_u = \mathcal{H}_{u-1} \cup \{h_u\}$.

Evidence-based Knowledge Validation. After exploration, the Explorer validates each hypothesis $h \in \mathcal{H}$ using the accumulated evidence $\mathcal{R}$, where $\mathcal{H}$ and $\mathcal{R}$ denote the final hypothesis set and evidence buffer, respectively. We consider only records $\mathcal{R}^{\mathrm{dir}}(h) \subseteq \mathcal{R}$ whose actions directly test h, while excluding prerequisite probes that do not provide direct evidence. We classify each record as *supporting* if the observed transition exhibits $\hat{o}_h$, *refuting* if it exhibits o_h^-, and *inconclusive* otherwise. To retain h as a reliable environment fact, we consider its supporting records $\mathcal{R}^+(h)$ and refuting records $\mathcal{R}^-(h)$, excluding inconclusive records. Specifically, $\mathcal{R}^+(h)$ must contain at least M_{sup} supporting records whose outcomes are reproduced in an independent exploration run under the same settings. When the action in h is applicable to multiple targets, these records must additionally cover at least N_{tgt} distinct targets to reduce instance-specific knowledge. We denote these requirements by $\mathrm{Pos}(h)$. We further require that $\mathcal{R}^-(h)$ contain no reproducible contradictory outcome, denoted by $\mathrm{NoNeg}(h)$. A hypothesis is retained only when both $\mathrm{Pos}(h)$ and $\mathrm{NoNeg}(h)$ hold. The resulting knowledge base $\mathcal{B}$ stores each validated action–outcome relation $(o_h, p_h, \hat{o}_h)$ together with its supporting evidence $\mathcal{R}^{\mathrm{sup}}(h)$:

$$\mathcal{B} = \{((o_h, p_h, \hat{o}_h),\, \mathcal{R}^{\mathrm{sup}}(h)) \mid h \in \mathcal{H},\ \mathrm{Pos}(h) \wedge \mathrm{NoNeg}(h)\}\,, \tag{2}$$

where $\mathcal{R}^{\mathrm{sup}}(h) \subseteq \mathcal{R}^+(h)$ denotes the supporting records retained as evidence for h.

2.3 Rubric-Guided Quality-Aware Skill Distillation

Online Trajectory Sampling. Rather than distilling a skill from a single trajectory, we compare multiple trajectories to identify behaviors to reuse or avoid. For each online task t, the Solver samples K trajectories, $\mathcal{T}_t = \{\tau_t^{(\ell)}\}_{\ell=1}^{K}$, conditioned on the task specification, initial observation, and relevant retrieval-active skills. Each trajectory is a sequence of actions and their resulting observations.

Rubric-Guided Trajectory Quality Assessment. To support skill distillation, we require a task-level signal that distinguishes high-quality trajectories from informative lower-quality trajectories without access to external outcome supervision. Towards this end, we employ a Judge to construct a task-specific, progress-aware rubric from the task specification and verified environment knowledge $\mathcal{B}$. The rubric contains n_t progress criteria that capture observable task progress and a goal criterion that captures complete task achievement. Figure 7 and Table 5 illustrate an example.

For robust quality assessment, the Judge independently evaluates each trajectory V times using the corresponding rubric. In the v-th evaluation, it outputs binary satisfaction indicators $y_{t,\ell,c}^{(v)} \in \{0,1\}$ for the c-th progress criterion and $y_{t,\ell,\mathrm{goal}}^{(v)} \in \{0,1\}$ for the goal criterion. When $n_t = 0$, the score reduces to goal satisfaction:

$$q_{t,\ell}^{(v)} = \begin{cases} \dfrac{1}{2n_t} \displaystyle\sum_{c=1}^{n_t} y_{t,\ell,c}^{(v)} + \dfrac{1}{2} y_{t,\ell,\mathrm{goal}}^{(v)}, & n_t > 0, \\ y_{t,\ell,\mathrm{goal}}^{(v)}, & n_t = 0. \end{cases} \tag{3}$$

The final quality score is the median over the V evaluations, $q_{t,\ell} = \mathrm{median}_v\, q_{t,\ell}^{(v)} \in [0,1]$.

Contrastive Skill Distillation. Based on the trajectory quality scores, SkillAdm distills two types of skills: *Action skills* that capture reusable procedures for advancing task execution, and *Avoidance skills* that capture behaviors that should be avoided. Following prior work on contrastive skill distillation (Zhao et al., 2024; Ma et al., 2026a), we construct two complementary trajectory sets: *positive trajectories*, which exhibit high quality, and *near-miss trajectories*, which achieve moderate quality while exhibiting behaviorally similar action patterns. Formally, the positive trajectory set is denoted as $\mathcal{T}_t^{\mathrm{pos}} = \left\{\tau_t^{(\ell)} \in \mathcal{T}_t : q_{t,\ell} \geq \gamma_+\right\}$, where γ_+ is the quality threshold. To identify near-miss trajectories, we represent each trajectory using normalized frequencies of action unigrams and bigrams, denoted by $z(\tau)$. The near-miss set is then defined as

$$\mathcal{T}_t^{\mathrm{nm}} = \left\{\tau_t^{(\ell)} \in \mathcal{T}_t \setminus \mathcal{T}_t^{\mathrm{pos}} : \gamma_{\mathrm{nm}} \leq q_{t,\ell} < \gamma_+,\ \cos\left(z(\tau_t^{(\ell)}), \bar{z}(\mathcal{T}_t^{\mathrm{pos}})\right) \geq \eta\right\}, \tag{4}$$

where $\bar{z}(\mathcal{T}_t^{\mathrm{pos}}) = \mathrm{Avg}_{\tau \in \mathcal{T}_t^{\mathrm{pos}}}\, z(\tau)$ denotes the average action-pattern representation of the positive trajectories. γ_{nm} and η denote the quality and similarity thresholds, respectively.

A Distiller contrasts $\mathcal{T}_t^{\text{pos}}$ and $\mathcal{T}_t^{\text{nm}}$ to identify behavioral differences associated with task progress. Action skill drafts are distilled from procedures present in positive trajectories but missing or incompletely executed in near-miss trajectories. Avoidance skill drafts are distilled from harmful or stalling behaviors present in near-miss trajectories but absent or appropriately handled in positive trajectories. Drafts describing similar behaviors are then consolidated into reusable skills. Skill distillation requires $\mathcal{T}_t^{\text{pos}} \neq \emptyset$ and produces only Action skills when $\mathcal{T}_t^{\text{nm}} = \emptyset$.

Finally, each newly distilled skill s comprises a skill type, a behavioral guideline specifying the target behavior to be tested, and an applicability description identifying the conditions under which the behavior is expected to be relevant. Let $\mathcal{E}(s)$ denote the supporting trajectory steps that provide the contexts motivating the skill. Each trajectory step consists of an action and its resulting observation.

2.4 Skill-Specific Utility Validation

Unlike existing work that relies on source tasks, previously collected tasks, or fixed offline validation splits, we construct executable skill-specific validation tasks tailored to each skill's applicable scenarios, directly measuring its utility where it is intended to apply.

Skill-Specific Validation Task Construction. This module takes three inputs: the skill description s (including its guideline and applicability description), its source evidence $\mathcal{E}(s)$, and verified environment knowledge $\mathcal{B}$. Based on these inputs, the Proposer generates N_{pro} validation-task proposals, each consisting of a task objective and a reproducible initial observation. A proposal is retained only if it satisfies three requirements: (a) the skill is applicable to the proposed scenario and its guidance can affect task execution; (b) the task's actions and expected outcomes are grounded in the environment description or verified knowledge $\mathcal{B}$; and (c) the task can be reproducibly initialized, executed through available environment actions, and evaluated from observable environment feedback. Let $\mathcal{V}_s^{\text{val}}$ denote the resulting set of validation tasks for skill s. If no proposal satisfies these requirements, skill s is discarded as its utility cannot be reliably assessed under available evidence.

Task-Level Rubric-Guided Skill Validation. Following the paired-comparison paradigm (Shang & Yang, 2026; Shang et al., 2026), we assess the utility of skill s by comparing matched Solver executions with and without s. For each validation task $t_i \in \mathcal{V}_s^{\text{val}}$, the with-skill condition provides s as-is, while the without-skill condition replaces it with a neutral placeholder containing no action guidance. All other task inputs and Solver configurations remain unchanged.

The Judge evaluates both trajectories using the same task-specific rubric constructed in Section 2.3. The trajectories are anonymized as X and Y to hide skill usage, and each is scored V times to reduce evaluation variance. Let $q_{t_i,\text{with}}^{(v)}(s)$ and $q_{t_i,\text{without}}^{(v)}(s)$ denote the corresponding scores. We define the task-level utility margin as the median paired score difference:

$$m_{t_i}^{\text{val}}(s) = \text{median}_{v=1,\ldots,V}\left[q_{t_i,\text{with}}^{(v)}(s) - q_{t_i,\text{without}}^{(v)}(s)\right]. \tag{5}$$

Significance-Aware Skill-Level Utility Validation. Given the task-level utility margins, a natural choice is to aggregate them by their mean across validation tasks as the overall utility score of skill s:

$$\overline{m}_s^{\text{val}} = \frac{1}{|\mathcal{V}_s^{\text{val}}|} \sum_{t_i \in \mathcal{V}_s^{\text{val}}} m_{t_i}^{\text{val}}(s). \tag{6}$$

The mean utility margin measures the average utility gain, but a positive mean alone does not indicate whether the gain is reliable. We therefore apply a one-sided Student's t-test to the task-level margins to test whether their mean is greater than zero, obtaining a p-value p_s^{val}. A smaller p_s^{val} provides stronger evidence of a positive average utility gain. We retain skill s only when both the utility gain and its statistical significance meet their respective thresholds:

$$A_s^{\text{val}} = \begin{cases} 1, & \overline{m}_s^{\text{val}} \geq \delta^{\text{val}} \wedge p_s^{\text{val}} \leq \alpha^{\text{val}}, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Here, δ^{val} and α^{val} are predefined thresholds for the mean utility margin and statistical significance, respectively. A skill with $A_s^{\text{val}} = 1$ passes validation and proceeds to future-task verification.

2.5 DECOUPLED ONLINE UTILITY VERIFICATION

The preceding module validates a skill on constructed skill-specific tasks. Inspired by prior work that reuses extracted skills to improve performance on subsequent tasks (Wang et al., 2025b; Xu et al., 2026), we further assess whether the validated utility transfers to naturally occurring future tasks. Specifically, we use these future tasks for verification before activating the skill for retrieval.

Applicability-Aware Future Verification Task Pre-Selection. Once skill s passes skill-specific utility validation ($A_s^{\text{val}} = 1$), we pre-select multiple future tasks that fall within its applicability for reliable verification. Specifically, future tasks are examined in arrival order. For each task t_j, we compute the textual similarity between its task specification and the applicability description of s, adding t_j to $\mathcal{V}_s^{\text{fut}}$ if the similarity exceeds the predefined threshold γ_{fut}. Selection stops when $|\mathcal{V}_s^{\text{fut}}| = N^{\text{fut}}$. If fewer than N^{fut} relevant tasks are available, skill s is discarded because its utility on future tasks cannot be reliably assessed.

Significance-Aware Skill-Level Utility Verification. Similar to the former validation stage, for each pre-selected task $t_j \in \mathcal{V}_s^{\text{fut}}$, we perform paired Solver executions with and without skill s. Both trajectories are evaluated using the shared task-specific rubric to obtain the task-level utility margin $m_{t_j}^{\text{fut}}(s)$ as in Equation (5). We then apply mean aggregation and the one-sided Student's t-test to these future-task margins to obtain the mean utility margin $\overline{m}_s^{\text{fut}}$ and p-value p_s^{fut}, respectively. Ultimately, the future-verification indicator is defined as:

$$A_s^{\text{fut}} = \begin{cases} 1, & \overline{m}_s^{\text{fut}} \geq \delta^{\text{fut}} \wedge p_s^{\text{fut}} \leq \alpha^{\text{fut}}, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

where δ^{fut} and α^{fut} are predefined thresholds, and $A_s^{\text{fut}} = 1$ indicates that skill s has passed future-task verification and is thus activated for retrieval.

3 EXPERIMENTS

We evaluate SkillAdm on three benchmarks, ALFWorld (Shridhar et al., 2021), ScienceWorld (Wang et al., 2022), and BabyAI (Chevalier-Boisvert et al., 2019), using three LLM backbones: Qwen3.6-35B (Qwen Team, 2026), Gemma-4-26B (Gemma Team, 2026), and GPT-OSS-120B (OpenAI, 2025). Our experiments address five research questions: (RQ1) task performance, (RQ2) component ablation, (RQ3) skill admission quality, (RQ4) rubric design, and (RQ5) cross-backbone skill transfer. Experimental details are provided in Appendix B.1.

3.1 TASK PERFORMANCE (RQ1)

Table 1 compares SkillAdm with No Skill, ASI (Wang et al., 2025a), the only oracle-free baseline, and the oracle-assisted baselines GRASP (Moll et al., 2026) and HDSO (Shang & Yang, 2026) across different benchmarks and LLM backbones. We observe the following: 1) SkillAdm consistently outperforms the oracle-free baseline ASI, with improvements of 2.75–17.51 points across the three benchmarks. 2) SkillAdm can outperform oracle-assisted baselines without relying on ground-truth outcomes on ScienceWorld and BabyAI. These results confirm the effectiveness of SkillAdm.

3.2 ABLATION STUDY (RQ2)

We compare SkillAdm with five variants on all three benchmarks using Qwen3.6-35B. **1) w/o-Rubric** replaces rubric-based trajectory evaluation with direct LLM judgment. **2) w/o-EnvKnow.** removes the verified environment knowledge $\mathcal{B}$ from rubric construction and skill-specific validation. **3) w/o-SkillVal.** removes skill-specific validation, retaining only future-task verification. **4) w/o-FutVerif.** removes future-task verification and retains only skill-specific validation. **5) w/o-SigCrit.** removes the significance criterion from both admission stages while retaining the mean utility-margin thresholds.

Table 2 reports ablation results. We observe the following: 1) Removing rubric-guided evaluation reduces performance across all benchmarks, highlighting the effectiveness of the proposed rubric-guided evaluation. 2) Removing environment knowledge also reduces performance, demonstrating its value for rubric construction and skill-specific validation. 3) Removing either skill-specific validation

Table 1: Task performance of different methods across benchmarks and LLM backbones.

Benchmark	Method	Qwen3.6-35B	Gemma-4-26B	GPT-OSS-120B
ALFWorld	No Skill	55.47	39.78	35.04
	GRASP	**79.56** ↑24.09	**74.45** ↑34.67	55.11 ↑20.07
	HDSO	73.36 ↑17.89	48.18 ↑8.40	48.54 ↑13.50
	ASI	65.69 ↑10.22	45.62 ↑5.84	43.07 ↑8.03
	SkillAdm	78.47 ↑23.00	53.65 ↑13.87	**60.58** ↑25.54
ScienceWorld	No Skill	20.81	11.87	21.03
	GRASP	30.74 ↑9.93	14.20 ↑2.33	31.15 ↑10.12
	HDSO	25.61 ↑4.80	**14.81** ↑2.94	25.63 ↑4.60
	ASI	24.53 ↑3.72	11.26 ↓0.61	25.02 ↑3.99
	SkillAdm	**32.16** ↑11.35	14.19 ↑2.32	**32.85** ↑11.82
BabyAI	No Skill	17.00	18.75	45.00
	GRASP	21.50 ↑4.50	**24.00** ↑5.25	52.00 ↑7.00
	HDSO	20.00 ↑3.00	21.50 ↑2.75	49.00 ↑4.00
	ASI	19.00 ↑2.00	19.00 ↑0.25	47.50 ↑2.50
	SkillAdm	**22.00** ↑5.00	21.75 ↑3.00	**53.50** ↑8.50

Orange : oracle-assisted methods, and blue : oracle-free methods. ↑/↓: absolute changes relative to No Skill.

Table 2: Ablation study with Qwen3.6-35B. ↓ denotes absolute decrease from Full.

Benchmark	Full	w/o-Rubric	w/o-EnvKnow.	w/o-SkillVal.	w/o-FutVerif.	w/o-SigCrit.
ALFWorld	**78.47**	74.09 ↓4.38	72.63 ↓5.84	72.99 ↓5.48	67.88 ↓10.59	66.42 ↓12.05
ScienceWorld	**32.16**	30.18 ↓1.98	29.51 ↓2.65	26.70 ↓5.46	27.36 ↓4.80	29.73 ↓2.43
BabyAI	**22.00**	20.50 ↓1.50	21.00 ↓1.00	19.00 ↓3.00	20.50 ↓1.50	19.00 ↓3.00

or future-task verification degrades performance, demonstrating the complementary value of the two admission stages. 4) Removing the significance criterion causes substantial degradation, indicating that mean utility margins alone cannot ensure consistent skill utility.

3.3 Skill Admission Quality (RQ3)

To assess skill admission quality, we report four metrics. 1) **Admission Rate (AR)** measures the fraction of generated skills that are admitted, reflecting admission selectivity. It is not inherently higher-is-better, as both over-admission and over-rejection can be undesirable. 2) **Useful Admission Rate (UAR)** measures the fraction of admitted skills that are useful, while **Useful Skill Recall (USR)** measures the fraction of useful generated skills that are retained. Together they characterize the usefulness and coverage of admitted skills. 3) **Harmful Admission Rate (HAR)** measures the fraction of admitted skills that are harmful, indicating the risk of retaining harmful skills.

Figure 3 shows the performance of SkillAdm and baselines regarding skill admission across three benchmarks using Qwen3.6-35B. As shown, SkillAdm achieves the highest USR and the lowest HAR across all three benchmarks, and the highest UAR on ScienceWorld and BabyAI, while remaining close to the oracle-assisted baseline GRASP on ALFWorld. This demonstrates that our oracle-free admission is effective and can even compete with oracle-assisted methods. Together with the strong UAR and USR and the lowest HAR, the lower AR further indicates that our method filters out harmful skills rather than discarding useful ones.

Figure 4 shows the ablation study results regarding skill admission across three benchmarks using Qwen3.6-35B. As shown, 1) Replacing rubrics with direct LLM judgment or removing environment knowledge lowers UAR and USR and raises HAR, indicating that explicit criteria and verified knowledge improve skill admission decisions. 2) Removing either admission stage increases HAR, highlighting the need to verify utility with both constructed skill-specific validation tasks and relevant future tasks. 3) Removing the significance criterion lowers UAR and increases HAR, showing that mean utility-margin thresholds alone yield less reliable admission.

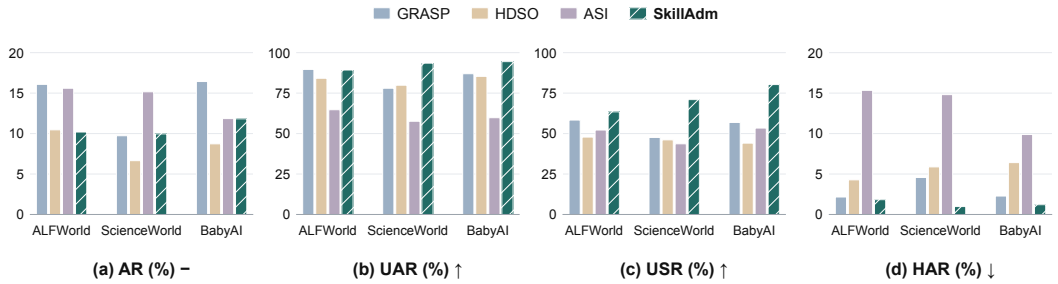

Figure 3: Skill admission quality of SkillAdm and baselines on three benchmarks with Qwen3.6-35B. AR is not higher-is-better, as both over-admission and over-rejection can be undesirable.

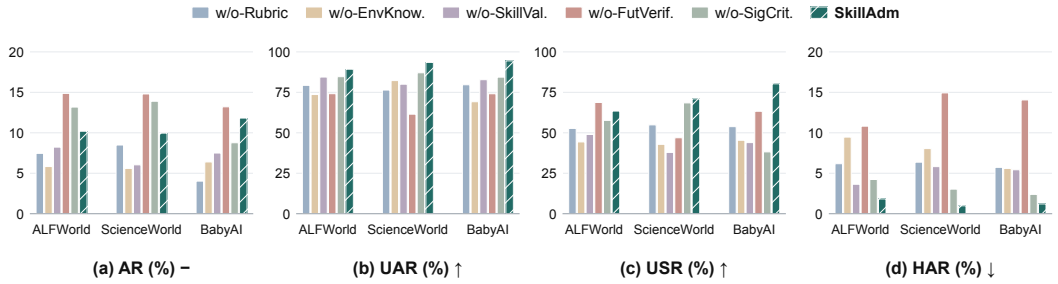

Figure 4: Skill admission quality of SkillAdm and variants on three benchmarks with Qwen3.6-35B.

3.4 Rubric Design (RQ4)

To evaluate the effectiveness of our designed rubric, we compare our SkillAdm with the following two variants. 1) **Direct-Judge** replaces rubric-based trajectory evaluation with direct LLM judgment. 2) **Goal-Only** removes the progress criteria while retaining the goal criterion.

For evaluation, we use a common metric across all benchmarks, along with benchmark-specific metrics tailored to their different outcome structures. *1) Common evaluation:* **Pairwise Accuracy (Pair Acc.)** measures whether trajectories with better benchmark outcomes receive higher trajectory scores. For each selected task, we randomly sample two trajectories with different benchmark outcomes to form a pair. Pair Acc. is computed as the average of these pairwise scores, assigning 1, 0, and 0.5 to correct, reversed, and tied evaluator rankings, respectively. *2) ALFWorld and BabyAI evaluation:* Since both benchmarks have binary task outcomes, we use **AUROC** to measure how well trajectory scores distinguish successful from failed trajectories. *3) ScienceWorld evaluation:* Since ScienceWorld provides graded task-completion scores that reflect partial progress even when a task is not fully completed, we use **Overall Spearman's ρ (Overall ρ)** to measure the rank correlation between trajectory scores and benchmark scores. We also report **Progress Spearman's ρ (Progress ρ)**, computed only over incomplete trajectories.

Table 3 shows the comparison results on the three benchmarks. As we can see, 1) compared with Direct-Judge, SkillAdm achieves substantially better performance across three benchmarks, implying the necessity of explicitly considering the progress criteria and the goal criterion in the rubric for trajectory ranking. 2) Goal-Only consistently outperforms Direct-Judge on ALFWorld and BabyAI, demonstrating the effectiveness of explicit goal-based criteria. However, it underperforms on ScienceWorld, particularly in Pair Acc. and Progress ρ (only 0.0410). This difference may stem from ScienceWorld's graded task-completion scores, which explicitly reflect partial progress even when a task is not fully completed. When evaluating the full trajectory, Direct-Judge may implicitly capture such intermediate progress to some extent, whereas Goal-Only focuses only on the final goal. 3) Our SkillAdm explicitly incorporates progress criteria, raising Progress ρ to 0.7060 and demonstrating the value of modeling intermediate progress for trajectory evaluation.

3.5 Cross-Backbone Skill Transfer (RQ5)

To assess whether skills can be reused across different LLM backbones, we examine the transferability of skill memory beyond the backbone that acquired the skills. Specifically, for each source backbone,

Table 3: Ablation results for different rubric designs across three benchmarks.

Evaluator	ALFWorld		ScienceWorld			BabyAI	
	Pair Acc.↑	AUROC ↑	Pair Acc.↑	Overall ρ ↑	Progress ρ ↑	Pair Acc.↑	AUROC ↑
Direct-Judge	88.16	0.8816	63.00	0.4470	0.1820	83.88	0.8388
Goal-Only	93.60	0.9360	60.00	0.5360	0.0410	85.25	0.8525
SkillAdm	**97.83**	**0.9742**	**79.33**	**0.7720**	**0.7060**	**94.75**	**0.9099**

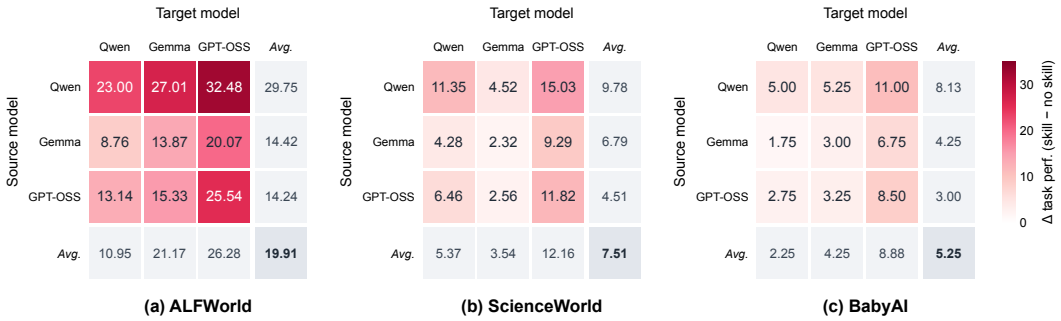

Figure 5: Cross-backbone skill transfer performance on three benchmarks. Rows: source backbones for skill memory construction; Columns: target backbones for task evaluation. Entries: gains over the target backbone without skills.

we first run SkillAdm to construct its skill memory and then freeze the resulting memory. We subsequently provide this frozen skill memory to each different target backbone, and evaluate the target Solver under identical execution settings.

Figure 5 visualizes the performance gains over the target's No Skill baseline across ALFWorld, ScienceWorld, and BabyAI. Two observations stand out. 1) For every source–target pair, transferring skill memory improves the target model over its No Skill baseline. This demonstrates that skills acquired by one LLM backbone can be effectively reused by other backbones, highlighting the cross-backbone transferability of skill memory. 2) Among the different source-target configurations, Qwen→GPT-OSS achieves a clear advantage across all three benchmarks, even outperforming both Qwen→Qwen and GPT-OSS→GPT-OSS. One possible explanation is that the larger-scale GPT-OSS-120B can more effectively exploit the structured task knowledge encoded in Qwen's skill memory. In contrast, GPT-OSS-120B already has stronger base capabilities, so its own skill memory may provide relatively smaller marginal benefits.

4 Conclusion

We introduce SkillAdm, an oracle-free framework for skill discovery and admission in frozen LLM agents. It uses verified environment knowledge to construct task-specific, progress-aware rubrics for trajectory evaluation and performs two-stage skill admission through validation on constructed tasks and independent verification on future tasks. Extensive experiments on three benchmarks and three LLM backbones show that SkillAdm consistently outperforms the unique oracle-free baseline ASI, while even outperforming oracle-assisted baselines in most settings. Further analyses show lower harmful admission rates and higher useful skill recall than both oracle-free and oracle-assisted baselines. The ablation study also confirms the effectiveness of each key component. Cross-backbone experiments further demonstrate that our skill memory generalizes across LLM backbones.

Limitations. SkillAdm requires paired executions and repeated evaluations to assess skill utility, introducing additional inference and environment interaction costs. Beyond these costs, admission decisions may also be affected by potential inaccuracies in the automatically generated rubrics and the subsequent LLM-based evaluations, since errors in either stage can propagate into the final admission outcome and lead to suboptimal skill memory. Future work could improve the efficiency and reliability of online test-time skill admission.

AI USE STATEMENT

We used generative AI tools to assist with manuscript drafting and language polishing, literature searches to identify relevant prior work, and code development. These tools assisted in drafting and revising the manuscript, identifying relevant references, and developing implementation code. The authors remain responsible for checking the accuracy and consistency of AI-assisted content and take full responsibility for the final manuscript, code, reported results, and scientific conclusions.

ETHICS STATEMENT

We study skill learning and admission in LLM agents using existing benchmark tasks and agent-generated trajectories in the simulated environments of ALFWorld, ScienceWorld, and BabyAI. These experiments involve neither human participants nor the collection of private personal data. Nevertheless, the skills and judgments generated by LLMs may contain errors or reflect model biases. Although our validation and verification procedures assess whether a skill improves task execution, they do not constitute a comprehensive safety assessment. Applying SkillAdm beyond these benchmarks would therefore require application-specific evaluation and safeguards.

REPRODUCIBILITY STATEMENT

Section 2 describes environment knowledge acquisition, skill distillation, validation-task construction, paired evaluation, and the statistical criteria for skill admission. Section 3 presents the baseline comparisons, ablation studies, rubric evaluation, and cross-backbone transfer protocol. Appendix B.1 provides the benchmark splits, evaluation metrics, model identifiers, interaction budgets, and hyper-parameter settings. Together, these descriptions provide the methodology, experimental settings, and evaluation protocols used in our experiments.

REFERENCES

Arthur Chen, Zuxin Liu, Jianguo Zhang, Akshara Prabhakar, Zhiwei Liu, Shelby Heinecke, Silvio Savarese, Victor Zhong, and Caiming Xiong. Test-time adaptation for LLM agents via environment interaction. In *Proceedings of the International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=OH4PE0TDo0.

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. AutoManual: Constructing instruction manuals by LLM agents via interactive environmental learning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 589–631, 2024. doi: 10.52202/079017-0019. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/0142921fad7ef9192bd87229cdafa9d4-Abstract-Conference.html.

Shiqi Chen, Jingze Gai, Ruochen Zhou, Jinghan Zhang, Tongyao Zhu, Junlong Li, Kangrui Wang, Zihan Wang, Zhengyu Chen, Klara Kaleb, Ning Miao, Siyang Gao, Cong Lu, Manling Li, Junxian He, and Yee Whye Teh. SkillCraft: Can LLM agents learn to use tools skillfully? *arXiv preprint arXiv:2603.00718*, 2026b. URL https://arxiv.org/abs/2603.00718.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. BabyAI: A platform to study the sample efficiency of grounded language learning. In *Proceedings of the International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJeXCo0cYX.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. AutoGuide: Automated generation and selection of context-aware guidelines for large language model agents. In *Advances in Neural Information Processing Systems*, volume 37, pp. 119919–119948, 2024. doi: 10.52202/079017-3811. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d8efbb5dd415974eb095c3f06bff1f48-Abstract-Conference.html.

Gemma Team. Gemma 4 Technical Report, 2026. URL https://arxiv.org/abs/2607.02770.

Jinyi Han, Yuanjian Xu, Ying Liao, Xinyi Wang, Zishang Jiang, Zixiang Di, Fanyang Lu, Zhichao Hu, and Yanghua Xiao. Skill-Use: Can LLMs actually use skills in agentic harnesses? *arXiv preprint arXiv:2608.04828*, 2026. URL https://arxiv.org/abs/2608.04828.

Jiaxi Li, Ke Deng, Yun Wang, Jingyuan Huang, Yucheng Shi, Qiaoyu Tan, Jin Lu, and Ninghao Liu. Online skill learning for web agents via state-grounded dynamic retrieval. *arXiv preprint arXiv:2606.04391*, 2026. URL https://arxiv.org/abs/2606.04391.

Hongjun Liu, Yifei Ming, Shafiq Joty, and Chen Zhao. Harnessing LLM agents with skill programs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2026. URL https://arxiv.org/abs/2605.17734.

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*, 2026a. URL https://arxiv.org/abs/2605.10999.

Ziyu Ma, Shidong Yang, Yuxiang Ji, Xucong Wang, Yong Wang, Yiming Hu, Tongwen Huang, and Xiangxiang Chu. SkillClaw: Let skills evolve collectively with agentic evolver. *arXiv preprint arXiv:2604.08377*, 2026b. URL https://arxiv.org/abs/2604.08377.

Johannes Moll, Jean-Philippe Corbeil, Jiazhen Pan, Martin Hadamitzky, Daniel Rueckert, Lisa Adams, and Keno Bressem. GRASP: Gated regression-aware skill proposer for self-improving LLM agents. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2026. URL https://arxiv.org/abs/2605.29668.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling agent self-evolving with reasoning memory. In *Proceedings of the International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jL7fwchScm.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, Tianjie Zhang, Wei Xu, Jie Tang, and Yuxiao Dong. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *Proceedings of the International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/c66e1fcc9691aae706250638f36f681b-Abstract-Conference.html.

Libin Qiu, Zhirong Gao, Junfu Chen, Yuhang Ye, Liangyu Li, Weizhi Huang, Xiaobo Xue, Wenkai Qiu, and Shuo Tang. AutoRefine: Compiling trajectories into validated typed agent artifacts. *arXiv preprint arXiv:2601.22758*, 2026. URL https://arxiv.org/abs/2601.22758.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

Fangxin Shang and Yehui Yang. Hypothesis-driven skill optimization for LLM agents. *arXiv preprint arXiv:2606.22330*, 2026. URL https://arxiv.org/abs/2606.22330.

Linfang Shang, Ming Xu, Yiding Sun, Tianle Xia, Lingxiang Hu, Lan Xu, and Ning Zheng. When self-evolution backfires: Pre-commit gating against skill contamination in LLM agents. *arXiv preprint arXiv:2608.05810*, 2026. URL https://arxiv.org/abs/2608.05810.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *Proceedings of the International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of LLM agents. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 7584–7600. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.409. URL https://aclanthology.org/2024.acl-long.409/.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.emnlp-main.775. URL https://aclanthology.org/2022.emnlp-main.775/.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks. In *Proceedings of the Conference on Language Modeling*, 2025a. URL https://openreview.net/forum?id=lsAY6fWsog.

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

Weijia Xu, Alessandro Sordoni, Chandan Singh, Zelalem Gero, Michel Galley, Xingdi Yuan, and Jianfeng Gao. Test-time learning with an evolving library. *arXiv preprint arXiv:2605.14477*, 2026. URL https://arxiv.org/abs/2605.14477.

Chen Yang, Jiashuo Tian, Ziqi Wang, Xinyin Liu, Meiru Ye, and Junjie Chen. Learning globally reusable skills for coding agents. *arXiv preprint arXiv:2608.06153*, 2026. URL https://arxiv.org/abs/2608.06153.

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

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling generalized agent abilities for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3053–3077. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.181. URL https://aclanthology.org/2024.findings-acl.181/.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, Xue Liu, Xiaoxiao Li, and Philip S. Yu. Co-EvoSkills: Self-evolving agent skills via co-evolutionary verification. In *Proceedings of the Conference on Language Modeling*, 2026a. URL https://arxiv.org/abs/2604.01687.

Qifan Zhang, Dongyang Ma, Tianqing Fang, Jia Li, Jing Tang, Nuo Chen, Haitao Mi, and Yan Wang. Training LLM agents for spontaneous, reward-free self-evolution via world knowledge exploration. *arXiv preprint arXiv:2604.18131*, 2026b. URL https://arxiv.org/abs/2604.18131.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *Proceedings of the International Conference on Learning Representations*, 2026c. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/8a94ff6f922d995d7d3f4ebf4143e442-Abstract-Conference.html.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025. URL https://arxiv.org/abs/2506.05176.

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. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Siyu Zhou, Tianyi Zhou, Yijun Yang, Guodong Long, Deheng Ye, Jing Jiang, and Chengqi Zhang. WALL-E: World alignment by neurosymbolic learning improves world model-based LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 72846–72885, 2025. doi: 10.52202/085713-2192. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/5e772a13ccba5255331240dcd99aa38b-Abstract-Conference.html.

Jiayin Zhu, Kelong Mao, Yudong Guo, Dengbo He, Sulong Xu, Simiu Gu, and Yutao Yue. Skill-Coach: Self-evolving rubrics for evaluating and enhancing agentic skill-use. *arXiv preprint arXiv:2607.01874*, 2026. URL https://arxiv.org/abs/2607.01874.

A Related Work

A.1 Memory-Based Agent Learning

Agents can learn reusable skills through interaction with the environment while keeping the underlying LLM fixed. Existing methods abstract execution trajectories into verbal reflections (Shinn et al., 2023), reusable insights (Zhao et al., 2024), behavioral guidelines (Fu et al., 2024), workflows (Wang et al., 2025b), or evolving manuals (Chen et al., 2024) for subsequent tasks. These forms of reusable knowledge are organized into a skill memory that can be expanded or revised using trajectories from later tasks (Chen et al., 2024; Zhang et al., 2026c; Xu et al., 2026; Li et al., 2026; Ma et al., 2026b). As this memory evolves, agents must decide which newly generated skills to retain for reuse. SkillAdm focuses on this admission decision: assessing whether a candidate skill improves task execution before making it available for retrieval.

A.2 Skill Validation and Evaluation

Existing methods assess newly distilled skills either by checking the skill itself or by evaluating task executions conditioned on the skill. Some code-oriented methods check syntax, interfaces, and executability before making a skill available for reuse (Chen et al., 2026b; Liu et al., 2026). These checks establish whether a skill can be executed, but not whether using it improves task execution. Assessing skill utility therefore requires two further choices: which tasks to use for validation and how to evaluate the resulting trajectories.

Validation tasks. Existing admission methods re-execute the task from which a skill was induced (Wang et al., 2025a), replay previously collected successful and failed cases (Moll et al., 2026; Yang et al., 2026; Qiu et al., 2026), or evaluate the skill on pre-allocated validation tasks (Shang & Yang, 2026; Shang et al., 2026). These methods obtain validation evidence from different task sources, and some can select tasks relevant to a new skill. However, their validation scenarios remain constrained by the available task pool: source-task re-execution and historical replay are tied to previously encountered situations, while a fixed validation pool may contain few tasks that exercise the capability targeted by a particular skill. SkillAdm addresses this limitation by first constructing executable skill-specific validation tasks from the skill, its source evidence, and verified environment knowledge, and then independently verifying whether the observed utility transfers to relevant naturally occurring future tasks.

Trajectory evaluation. Some methods evaluate task executions using externally provided ground-truth outcomes (Shang & Yang, 2026; Moll et al., 2026; Zhang et al., 2026a). ASI (Wang et al., 2025a) avoids this requirement during admission by using an LLM evaluator to directly judge task correctness, but its binary correctness signal provides limited information about different levels of progress among unsuccessful executions. Recent work instead uses rubric-based evaluation to provide finer-grained assessments of skill use and execution (Zhu et al., 2026; Han et al., 2026). However, these rubrics primarily assess whether the provided skill is selected or executed appropriately, which does not necessarily establish progress toward the task goal. SkillAdm instead constructs task-specific, progress-aware rubrics that assess intermediate task progress and final goal completion, and applies the same fixed rubric to matched executions with and without the skill.

B Experimental Details

B.1 Experiment Setup

Datasets. We evaluate SkillAdm on three interactive benchmarks: ALFWorld (Shridhar et al., 2021), ScienceWorld (Wang et al., 2022), and BabyAI (Chevalier-Boisvert et al., 2019). Table 4 summarizes four aspects of each benchmark: *Domain*, the task domain; *Exploration*, the number of tasks used for pre-task environment knowledge acquisition; *Learning*, the number of tasks used for online skill learning and admission; and *Held-out*, the number of tasks used for final evaluation with the learned skill memory frozen. Learning tasks for ALFWorld and ScienceWorld are sampled proportionally by task type from their official training splits. BabyAI instead provides a procedural task generator, which we use to construct separate learning and held-out task sets.

Table 4: Benchmark overview.

Benchmark	Domain	Exploration	Learning	Held-out
ALFWorld	Household tasks	1,515	1,024	274
ScienceWorld	Scientific reasoning	1,012	1,024	150
BabyAI	Instruction following	640	600	400

Evaluation metrics. We report task performance using the standard evaluation metric of each benchmark (Shridhar et al., 2021; Wang et al., 2022; Chevalier-Boisvert et al., 2019). We additionally report the following metrics to assess skill admission quality. 1) **Admission Rate (AR)** measures the fraction of generated skills that are admitted, reflecting the selectivity of the admission process. AR is not inherently a higher-is-better metric, since both overly aggressive and overly conservative admission can be undesirable. 2) **Useful Admission Rate (UAR)** measures the fraction of admitted skills that are labeled useful, while **Useful Skill Recall (USR)** measures the fraction of all useful generated skills that are admitted. Together, they characterize the usefulness of the admitted skill set and the ability to retain useful skills. 3) **Harmful Admission Rate (HAR)** measures the fraction of admitted skills that are harmful, indicating how often harmful skills are retained.

To compute these metrics, we label each generated skill according to its effect on task performance. Specifically, for each generated skill, we identify applicable tasks by comparing embeddings (Zhang et al., 2025) of its applicability description and task objectives, retaining tasks with similarity above 0.55. Each selected task is executed with and without the skill under otherwise identical conditions. We measure the task-level performance gain using official benchmark outcomes and average these gains across applicable tasks. To account for uncertainty in the estimated mean gain, we apply bootstrap resampling over the applicable tasks to construct a 95% confidence interval. A skill is labeled *useful* if the lower confidence bound is above zero and *harmful* if the upper confidence bound is below zero. Otherwise, it is labeled *inconclusive*, including when no applicable tasks are identified.

Baselines. To evaluate SkillAdm, we compare SkillAdm with four baselines grouped into three categories. 1) **No Skill.** This baseline uses the same frozen Solver with an empty skill memory and performs neither skill retrieval nor skill admission. 2) **Oracle-free skill admission.** We include ASI (Wang et al., 2025a), which re-executes the inducing task with each newly generated programmatic skill and admits the skill only when the execution passes source-task verification, without relying on externally provided validation outcomes. 3) **Oracle-assisted skill admission.** We additionally include oracle-assisted methods as stronger reference points to assess how closely our oracle-free SkillAdm can approach methods with access to additional validation supervision. Specifically, GRASP (Moll et al., 2026) evaluates proposed skill updates by replaying previously successful and failed cases, while HDSO (Shang & Yang, 2026) evaluates skills on a pre-allocated validation task pool before admission.

Implementation details. We use three frozen LLM backbones: Qwen3.6-35B-A3B-FP8 (Qwen Team, 2026), Gemma-4-26B-A4B-it (Gemma Team, 2026), and GPT-OSS-120B (OpenAI, 2025). During skill learning, the selected backbone is shared by the Explorer, Solver, Distiller, Judge, and Proposer. We use the same hyperparameter configuration across all three benchmarks. Specifically, we use Qwen3-Embedding-0.6B (Zhang et al., 2025) for similarity-based skill retrieval. For *Pre-Task Environment Knowledge Acquisition*, we allow at most 32 probe iterations per exploration run and set the minimum supporting-evidence and target-coverage requirements to $M_{\text{sup}} = 3$ and $N_{\text{tgt}} = 2$, respectively. For *Rubric-Guided Quality-Aware Skill Distillation*, the Solver samples $K = 5$ trajectories for each online task, with each trajectory limited to 50 Agent–Environment interaction turns. The Judge independently evaluates each trajectory $V = 3$ times. We set the positive-trajectory and near-miss quality thresholds to $\gamma_{+} = 0.60$ and $\gamma_{\text{nm}} = 0.20$, respectively, and the action-pattern similarity threshold to $\eta = 0.80$. Then for *Skill-Specific Utility Validation*, the Proposer generates $N_{\text{pro}} = 12$ validation-task proposals for each candidate skill. We set the validation utility-margin and significance thresholds to $\delta^{\text{val}} = 0.03$ and $\alpha^{\text{val}} = 0.05$, respectively. Ultimately, for *Decoupled Online Utility Verification*, we select future tasks using the relevance threshold $\gamma_{\text{fut}} = 0.55$ and set the required number of relevant tasks as $N^{\text{fut}} = 80$. We set the corresponding utility-margin and significance thresholds to $\delta^{\text{fut}} = 0.05$ and $\alpha^{\text{fut}} = 0.05$.

Following ReasoningBank (Ouyang et al., 2026), we initialize all methods with an empty skill memory to evaluate skill acquisition through online interaction. We keep the task order, environment initialization, maximum number of environment steps per task, and Solver execution settings identical across methods. After online learning, we freeze each method's skill memory and evaluate all methods on the same held-out task instances.

C Sensitivity Analysis

We investigate the sensitivity of SkillAdm to three hyperparameters governing the selection of positive and near-miss trajectories for contrastive skill distillation: the positive-trajectory quality threshold γ_+, the near-miss quality threshold γ_{nm}, and the action-pattern similarity threshold η. Their default values are provided in Appendix B.1. Given the computational cost of evaluating all configurations across multiple environments, we conduct the sensitivity analysis on ALFWorld, varying one hyperparameter at a time while keeping all others fixed.

On γ_+. We vary $\gamma_+ \in \{0.50, 0.60, 0.70, 0.80, 0.90\}$. As shown in Figure 6(a), performance remains relatively stable when γ_+ ranges from 0.50 to 0.70, but decreases at the higher values of 0.80 and 0.90. A higher threshold imposes a stricter quality requirement on positive trajectories and may therefore leave fewer high-quality examples for subsequent skill distillation.

On γ_{nm}. We vary $\gamma_{\text{nm}} \in \{0.10, 0.20, 0.30, 0.40\}$. Figure 6(b) shows that performance is relatively stable at lower thresholds and declines as γ_{nm} increases beyond the default value of 0.20. Increasing γ_{nm} filters out more lower-scoring trajectories, potentially reducing the number of informative near-miss examples available for contrastive comparison. This suggests that retaining moderately lower-quality but behaviorally related trajectories is useful for identifying behaviors that distinguish better executions from worse ones.

On η. We vary $\eta \in \{0.20, 0.35, 0.50, 0.65, 0.80, 0.95\}$. As shown in Figure 6(c), the effect of η is non-monotonic: performance first decreases, then recovers to its highest value at the default setting $\eta = 0.80$, before dropping sharply at $\eta = 0.95$. Selecting near-miss trajectories with action patterns similar to those of positive trajectories helps the Distiller identify missing or incorrect actions. However, setting η too high may leave too few near-miss examples for skill distillation.

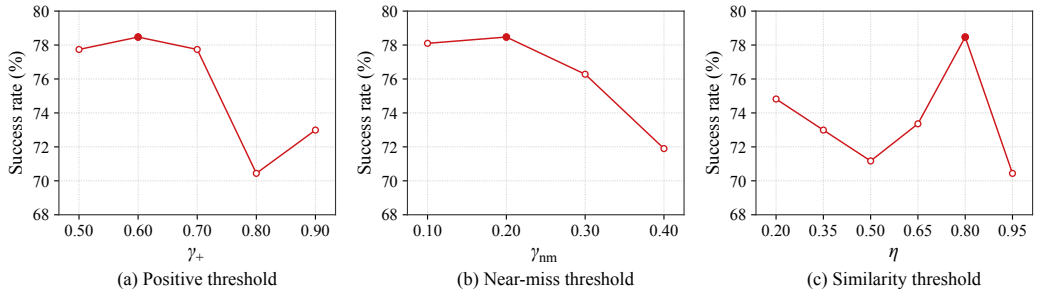

Figure 6: Sensitivity of SkillAdm on ALFWorld to (a) the positive-trajectory quality threshold γ_+, (b) the near-miss quality threshold γ_{nm}, and (c) the action-pattern similarity threshold η.

D Case Studies

In this section, we provide an illustrative case, adapted from an ALFWorld learning task, showing how positive and near-miss trajectories support contrastive skill distillation and how the resulting skill undergoes skill-specific utility validation and future-task verification.

Positive and Near-Miss Trajectories. Figure 7 shows a positive trajectory and a near-miss trajectory for the same task of *put a hot tomato in the fridge*, starting from the same initial environment state and with the same action space. Due to space constraints, we only list the first step, the intermediate steps where the two trajectories diverge, and the final step; the remaining intermediate parts are largely similar. As can be seen, the divergence occurs during the intermediate interaction: in the **positive trajectory**, the Solver correctly opens the microwave and explicitly heats the held tomato before

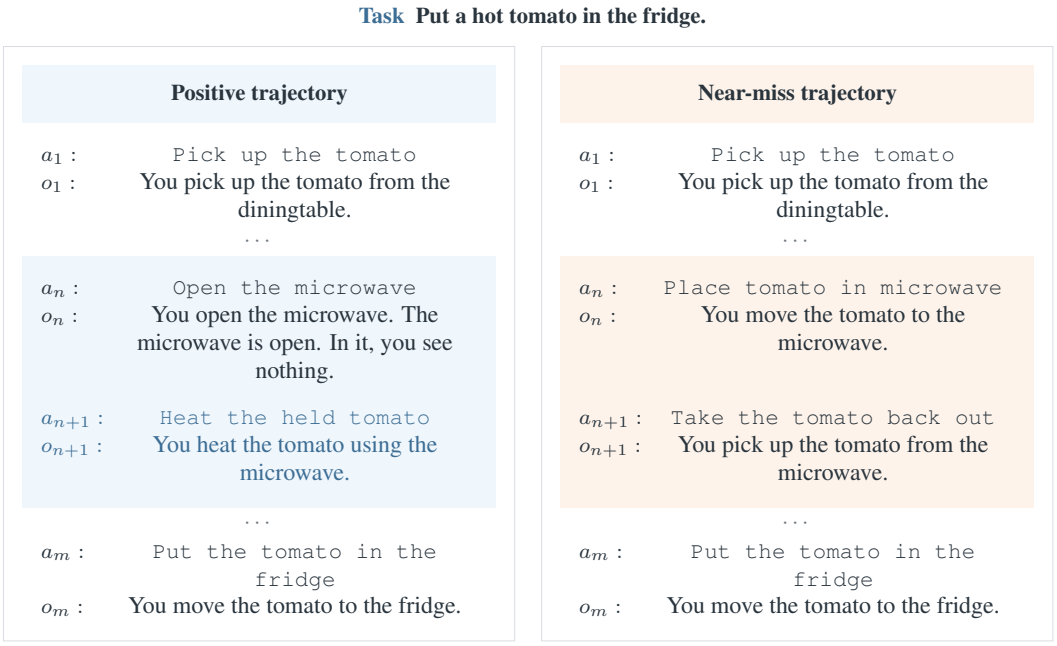

Figure 7: Positive and near-miss trajectories for the task of *placing a hot tomato in the fridge*.

placement, whereas in the **near-miss trajectory**, the Solver puts the tomato inside the microwave and later retrieves it without executing the heating action, thereby failing to make the tomato hot. This contrast suggests that our near-miss trajectory filtering method is effective, as it yields trajectory pairs that are directly comparable and thus provides a clean contrastive signal for skill extraction.

Task-Specific Progress-Aware Rubric. Given the aforementioned task specification and the verified environment knowledge $\mathcal{B}$, the Judge derives a task-specific, progress-aware rubric. As shown in Table 5, this case yields three progress criteria, assessing whether the tomato is acquired, heated, and placed in the fridge, respectively, together with a goal criterion requiring the tomato to be heated before placement. Overall, these criteria cover the essential steps and final goal of the task, showing that the rubric is well aligned with the task.

Table 5: Task-specific progress-aware rubric for *put a hot tomato in the fridge*.

Criterion	Type
Tomato acquired	Progress
Tomato is hot	Progress
Tomato placed in fridge	Progress
Hot tomato placed in fridge	Goal

Contrastive Skill Distillation. The near-miss trajectory puts the tomato into the microwave and retrieves it without heating, whereas the positive trajectory explicitly executes the `heat` action on the held tomato. The Distiller turns this difference into a reusable instruction: execute the heating action before placing the target at its destination. The specific tomato and fridge are then generalized to an object and a destination, yielding the reusable skill shown in Figure 8.

Constructed Skill-Specific Utility Validation Task. Given the Action skill in Figure 8, its source evidence $\mathcal{E}(s)$, and verified environment knowledge $\mathcal{B}$, the Proposer constructs validation tasks involving the `heat` action. Figure 9 shows one such task, whose objective is *put a hot mug in dining table* and whose initial observation describes a mug on a dining table, with a microwave available. This task matches the applicability of the above action skill and requires the same behavior as its guideline, showing that the constructed task is well aligned with the skill to be validated.

Action Skill

Applicability
Tasks that require placing a heated object at a specified destination, with a microwave available for heating.

Guideline
While holding the target, execute the `heat` action using the microwave, then place the heated object at the specified destination.

Figure 8: An Action skill distilled from the positive–near-miss trajectory contrast.

Constructed validation task

Task objective
put a hot mug in dining table.

Initial observation
A mug is on a dining table; a microwave is available.

Figure 9: A constructed skill-specific utility validation task from this case.

Relevant Future Verification Task. After the Action skill passes skill-specific validation, we select relevant future tasks in arrival order based on the textual similarity between their task specifications and the skill's applicability description. Figure 10 shows one such task, whose objective is *heat some cup and put it in shelf* and whose initial observation indicates that a cup and a microwave are available, with a shelf as the destination. This task follows the same heat-then-place structure as the skill's applicability, while involving a different object and destination. It is therefore a relevant future task for verifying whether the validated skill transfers to naturally occurring tasks.

Relevant future task

Task objective
heat some cup and put it in shelf.

Initial observation
A cup and a microwave are available; the destination is a shelf.

Figure 10: A relevant future verification task from this case.