# SkillScaffold: Scaffolding Agentic Reinforcement Learning with Policy-Adaptive Skill Credit Assignment and Selective Injection

**Anonymous authors**
Paper under double-blind review

## Abstract

Agentic reinforcement learning (RL) enables large language models (LLMs) to learn from multi-turn interactions with verifiable outcomes. Textual skills can improve exploration by distilling prior interactions into reusable guidance. However, existing skill-based RL approaches often either assume skill injection is beneficial or rely on indirect feedback or rollout-intensive evaluation to assess its utility. Our pilot studies show that rollout-based utility estimation for skill credit assignment is costly and unstable under limited sampling, while skill benefits vary across candidates and depend on the current policy's competence on each query. To address these challenges, we introduce SkillScaffold, a policy-adaptive training scaffold that combines low-cost skill-utility estimation with selective skill injection. By reusing outcome-verified training trajectories, SkillScaffold generates candidate skills and estimates their utility through outcome-aware likelihood comparisons, without candidate-specific environment rollouts. The resulting utility guides whether and which skill to inject, retaining skill-free execution when no candidate is estimated to help. Joint training enables the Generator to produce skills tailored to the evolving Executor. Scaffolded RL encourages skill internalization for skill-free inference. Experiments on ALFWorld, WebShop, ScienceWorld, and $\tau^2$-Bench demonstrate gains over strong agentic-RL and skill-based RL baselines across both short- and long-horizon tasks. Ablations and further analyses support the importance of utility-based selection and abstention.

## 1 Introduction

Agentic reinforcement learning (RL) enables language-model agents to improve through multi-turn interaction with verifiable environments, but long horizons and sparse outcome rewards make successful behavior difficult to discover and reinforce (Shao et al., 2024; Dong et al., 2025). Textual skills can ease this challenge by compressing trajectories or feedback into reusable procedural guidance for subsequent decisions (Shinn et al., 2023; Zhao et al., 2024). Existing approaches extract, refine, and retrieve such guidance using external memory or teacher models (Nottingham et al., 2024; Fu et al., 2024; Chen et al., 2024; Wang et al., 2024; Zhou et al., 2026; Fang et al., 2026). SkillRL, for example, uses a stronger teacher to build an evolving SkillBank, turning sparse trajectory-level feedback into guidance for subsequent exploration (Xia et al., 2026).

Recent work shows that a shared policy can generate skills from experience and use them for subsequent problem solving, allowing generation and execution capabilities to co-evolve (Wang et al., 2026b; Xia et al., 2026; Shi et al., 2026; Fan et al., 2026; Zhou et al., 2026; Zhang et al., 2026; He et al., 2026). Complementary work internalizes skill-conditioned behavior via curriculum learning or self-distillation, enabling skill-free inference (Lu et al., 2026; Wang et al., 2026a). Building on these directions, we study skills as *policy-adaptive training scaffolds*, focusing on evaluating and selectively using guidance for the current policy and query.

Under this training-time scaffold view, *skill credit assignment* uses a scalar *skill utility* for each candidate to determine whether any skill should be used and which one to select. Existing approaches either use generated skills without testing their benefit (Xia et al., 2026; Zhang et al., 2026), rely on indirect and delayed feedback (Shi et al., 2026; Zhang et al., 2026), or estimate utility through



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.

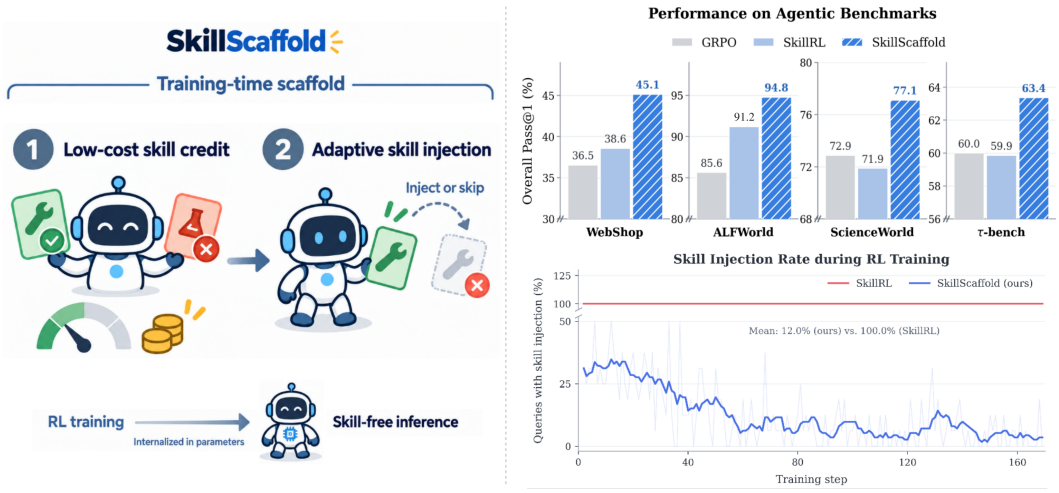



**Figure 1. Overview and empirical summary of SkillScaffold.** Low-cost, policy-relative skill-utility estimates guide selective skill injection during training, while inference remains skill-free. Across four agentic benchmarks, SkillScaffold consistently outperforms GRPO and SkillRL while injecting skills for only a small fraction of training queries.

additional skill-conditioned environment rollouts (Fan et al., 2026; He et al., 2026; Shen et al., 2026). Our pilot studies show that rollout-based estimates are costly and unstable under limited sampling, while skill benefits vary across candidates and with how reliably the current policy solves each query without skills (Dong et al., 2026). These findings motivate a low-cost, policy-relative utility signal for candidate ranking and abstention.

To address these challenges, we propose SkillScaffold, a policy-adaptive training scaffold for agentic RL, summarized in Figure 1. Its core idea is to replace candidate-specific environment evaluation with outcome-aware likelihood comparisons on existing training trajectories. A shared policy serves as both an *Executor*, which collects skill-free Scout trajectories, and a *Generator*, which produces candidate skills from outcome-labeled Scout evidence. For each candidate, SkillScaffold rescores fixed Scout evidence under candidate-skill and no-skill contexts to obtain a policy-relative utility estimate. This proxy favors increased likelihood of successful behavior and reduced likelihood of observed failures, according to the available outcome evidence. The highest-utility candidate guides fresh Executor rollouts only when its estimated utility is positive; otherwise, execution remains skill-free. Environment rewards supervise the Executor, while skill utility supervises the Generator through role-specific GRPO objectives, enabling the Generator to adapt its guidance to the evolving Executor.

Skill-free Scouts and selectively scaffolded Executor trajectories jointly update the shared policy, encouraging skill internalization. At deployment, the Generator and the entire skill pathway are removed. We evaluate SkillScaffold on ALFWorld, WebShop, ScienceWorld, and $\tau^2$-Bench, covering diverse multi-turn environments. Across these settings, SkillScaffold improves over strong agentic-RL and skill-based RL baselines under fully skill-free inference. Ablations and further analyses support the importance of utility-based selection and abstention.

In summary, the key contributions of this work are as follows:

- We identify two key challenges in skill credit assignment: rollout-based evaluation is costly and unstable, while skill benefits depend on both the candidate and how reliably the current policy can solve the query without skills, motivating candidate ranking and abstention.
- We propose SkillScaffold, a shared-policy training scaffold that estimates candidate utility through outcome-aware likelihood comparisons on existing trajectories. The resulting signal guides skill selection, abstention, and Generator learning without candidate-specific environment rollouts.
- Experiments on ALFWorld, WebShop, ScienceWorld, and $\tau^2$-Bench demonstrate improvements over strong agentic-RL and skill-based RL baselines under skill-free inference. Ablations and further analyses support the importance of utility-based selection and abstention.

## 2 Preliminaries

**Agentic reinforcement learning with skills.** We consider an LLM policy $\pi_\theta$ that interacts with an environment over multiple turns through a tool engine $\mathcal{T}$. Given a task $x \sim \mathcal{D}$ and an optional textual skill $s$, the Executor generates an interaction trajectory $\tau = (a_0, o_0, \ldots, a_K, o_K)$ conditioned on $(x, s)$, where

$$a_t \sim \pi_\theta^{\text{exec}}(\cdot \mid x, s, \tau_{<t}; \mathcal{T}), \qquad o_t = \mathcal{T}(a_t). \tag{1}$$

Here $s = \varnothing$ denotes skill-free execution. A rule-based Outcome Verifier assigns an outcome reward $R(\tau; x)$, and the Executor is optimized to maximize expected environment return:

$$\max_\theta J_{\text{env}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, \tau\sim\pi_\theta^{\text{exec}}(\cdot|x,s;\mathcal{T})}[R(\tau; x)]. \tag{2}$$

**Group Relative Policy Optimization (GRPO).** GRPO (Shao et al., 2024) optimizes the policy using group-relative advantages. For each task $x$, a group of $G$ trajectories $\{\tau_i\}_{i=1}^G$ is sampled from the old policy, and the policy is updated with

$$J_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\tau_i|}\sum_{t=1}^{|\tau_i|}\left[\min\left(\rho_{i,t}\widehat{A}_i, \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon)\widehat{A}_i\right) - \beta D_{\text{KL}}^{i,t}\right]\right]. \tag{3}$$

Here $\rho_{i,t}$ denotes the token-level importance ratio, $\widehat{A}_i$ is the group-normalized trajectory advantage, $\epsilon$ is the clipping threshold, and $\beta$ controls the KL penalty.

## 3 Motivating Observations

Skill-based agentic RL must decide whether and which generated skill should guide subsequent exploration. Rollout-based utility estimation is costly and unstable under practical sampling budgets, while skill utility varies across candidates and with the current policy. We first characterize these two challenges before introducing SkillScaffold.

### 3.1 Rollout-Based Skill-Utility Estimation Is Unstable and Costly

For a task $x$ with $M$ candidate skills $C_x = \{s_i\}_{i=1}^M$, direct skill utility is the downstream improvement over skill-free execution:

$$U_{\text{DR}}(s; x) = \mathbb{E}_{\tau\sim\pi_\theta^{\text{exec}}(\cdot|x,s;\mathcal{T})}[R(\tau; x)] - \mathbb{E}_{\tau\sim\pi_\theta^{\text{exec}}(\cdot|x,\varnothing;\mathcal{T})}[R(\tau; x)]. \tag{4}$$

Estimating each condition with $K$ episodes and sharing the skill-free baseline requires $(M + 1)K$ additional environment rollouts per query.

**Pilot observation.** For each candidate–query pair and budget $K$, we form two independent estimates of $U_{\text{DR}}$ and record a sign disagreement when both are non-zero but have opposite signs. Figure 2(a) shows that disagreement decreases only from 23% at $K = 1$ to 16% at $K = 8$ under sparse outcome rewards.

A query-level candidate group is non-tied when its $M$ candidates do not all receive the same empirical utility. Figure 2(b) shows that increasing the additional rollout budget eight-fold, from 5 to 40 per query, raises the fraction of non-tied groups only from 32% to 55%, leaving nearly half unresolved.

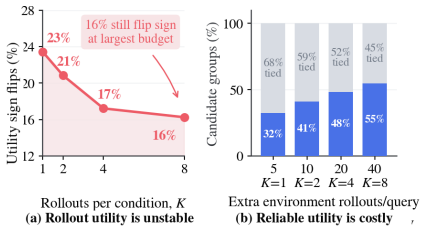



**Figure 2. Rollout-based utility estimation is costly and unstable.** (a) Sign disagreement between independent estimates. (b) More rollout samples distinguish more candidate groups.

Online RL must immediately decide *whether* and *which* skill to inject, whereas reliable candidate-specific estimates require additional interaction. We therefore seek a low-cost utility estimate that avoids candidate-specific downstream rollouts.

### 3.2 Skill Utility Is Query- and Policy-Dependent

Low interaction cost alone is insufficient: skill utility varies across queries, candidates, and policy competence, so credit assignment must both rank candidates and abstain when none is helpful.

**Within-query variation.** On a non-saturated pilot subset where candidate differences remain measurable, Figure 3(a) compares the best and worst observed candidates for each query, as measured by downstream rollouts relative to skill-free execution. Among 100 queries, 28 contain both positive- and negative-effect candidates, showing that generating plausible skills is insufficient; candidates must be distinguished at the query level.

**Dependence on policy competence.** We measure current competence using four independent skill-free probe rollouts from the same frozen policy. Here, a probe is an evaluation-only rollout used to assess the policy's current skill-free competence. This avoids conflation with historical training-time Scout outcomes. We group queries by their number of successful probes (0–4). Figure 3(b) reports both the mean candidate effect and the best observed candidate. Average skill gains are positive when only a minority of probes succeed; when most or all succeed, the mean effect becomes negative and even the best observed candidate provides little or no additional benefit.

Thus, skill credit assignment requires a *query- and policy-relative utility* to identify the most helpful candidate and abstain when none provides positive additional value. These downstream measurements are used only for diagnosis, not as training signals for SkillScaffold.

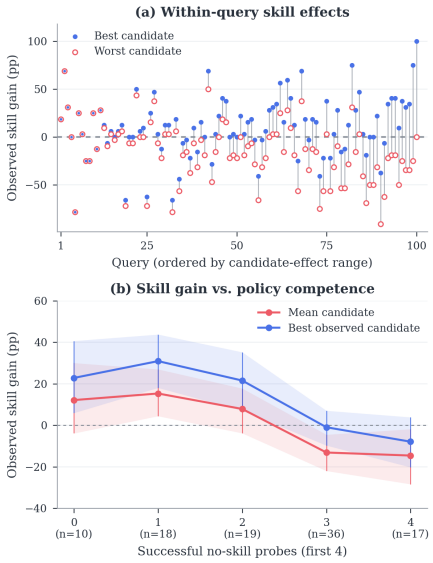



**Figure 3. Skill utility is query- and policy-dependent.** (a) Candidate effects vary within a query. (b) Skill benefit decreases as skill-free policy competence improves.

## 4 SkillScaffold

SkillScaffold uses one shared policy $\pi_\theta$ in two roles: the *Executor* interacts with the environment, while the *Generator* produces candidate skills from outcome-labeled trajectories. As shown in Figure 4, each update collects skill-free Scouts, generates and scores candidate skills, and injects only the highest-utility positive candidate into fresh Executor rollouts. Environment rewards train the Executor, skill utility trains the Generator, and their shared parameters enable fully skill-free deployment.

### 4.1 Skill-Free Rollout

**Skill-free Scout collection.** For task $x$, the current Executor samples $N$ skill-free trajectories:

$$\tau_i^0 \sim \pi_\theta^{\text{exec}}(\cdot \mid x, \varnothing; \mathcal{T}), \qquad i = 1, \ldots, N. \tag{5}$$

We call these trajectories *Scouts*. An environment verifier assigns each Scout an outcome $y_i \in \{0, 1\}$, which defines $\mathcal{S}_x = \{i : y_i = 1\}$ and $\mathcal{F}_x = \{i : y_i = 0\}$. The resulting group is all-success, all-failure, or mixed.

### 4.2 Outcome-Anchored Skill

**Group-based evidence.** The Generator receives the task and all outcome-labeled Scouts. It extracts reusable decisions from all-success groups, diagnoses recurring errors from all-failure groups, and contrasts successful and failed behavior in mixed groups. Thus, both successes and failures provide query-specific evidence for skill generation.

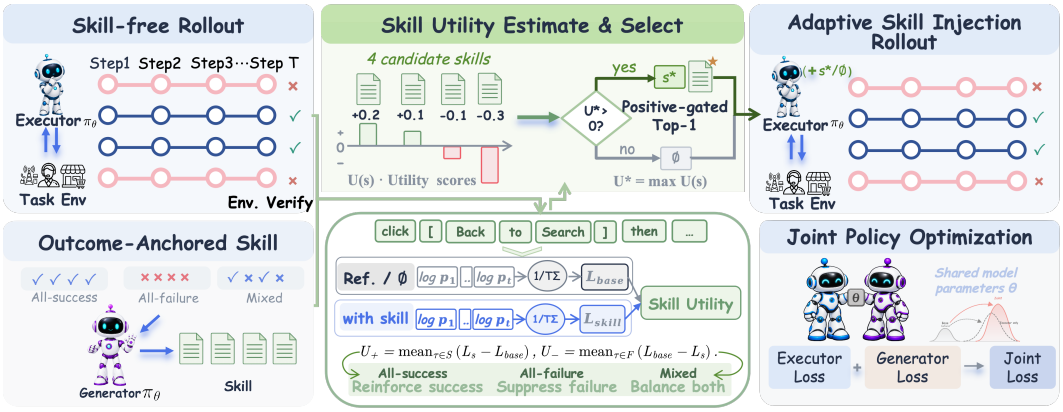



**Figure 4. Overview of SkillScaffold.** The five stages show skill-free scouting, outcome-anchored generation, utility-based selection, adaptive injection, and joint policy optimization.

**Anchor-based evidence.** We also consider a lightweight variant that uses one verified successful Scout as its evidence anchor. When $\mathcal{S}_x \neq \varnothing$, we choose $a_x = \min \mathcal{S}_x$ and provide $\tau^0_{a_x}$ to the Generator. If no Scout succeeds, this variant skips generation and keeps the subsequent Executor rollouts skill-free.

**Candidate generation.** Let $E^v_x$ denote the evidence record for variant $v \in \{\text{grp}, \text{anc}\}$. The Generator independently samples $M$ candidate skills,

$$s_j \sim \pi^{\text{gen}}_\theta(\cdot \mid E^v_x), \qquad C_x = \{s_j\}_{j=1}^M. \tag{6}$$

Each candidate is a structured skill card containing applicability conditions, an executable procedure, conditional decision rules, and verification, recovery, or stopping criteria. Format-invalid outputs are excluded from selection.

### 4.3 Skill Utility Estimate & Select

A skill credit-assignment mechanism must assign each candidate a scalar *skill utility* $U(s; x)$. A useful skill should be compatible with verified successful behavior and incompatible with observed failed behavior. Directly testing this criterion with fresh environment rollouts would reproduce the cost and variance identified in Section 3.1. Instead, we keep the Scouts fixed and compare how the current policy scores the same behavior with and without each candidate skill.

**Paired trajectory scoring.** Write the Executor-generated reasoning and action tokens in a fixed Scout trajectory $\tau$ as $u_{1:T}$. For skill context $c$, define

$$\ell_\theta(c; \tau) = \frac{1}{T} \sum_{t=1}^{T} \log \pi^{\text{exec}}_\theta(u_t \mid x, c, \tau_{<t}), \qquad d(s, \tau) = \ell_\theta(s; \tau) - \ell_\theta(\varnothing; \tau). \tag{7}$$

Here $\tau_{<t}$ is the recorded interaction prefix preceding $u_t$, including previous Executor outputs and environment observations. A positive $d(s, \tau)$ means that conditioning on $s$ increases the likelihood of the observed Scout behavior relative to skill-free execution.

**Group-based utility.** We aggregate the trajectory-level differences according to the observed outcomes:

$$U^{\text{grp}}(s; x) = \begin{cases} \dfrac{1}{|\mathcal{S}_x|} \displaystyle\sum_{i \in \mathcal{S}_x} d(s, \tau^0_i), & \mathcal{F}_x = \varnothing, \\ -\dfrac{1}{|\mathcal{F}_x|} \displaystyle\sum_{i \in \mathcal{F}_x} d(s, \tau^0_i), & \mathcal{S}_x = \varnothing, \\ \dfrac{1}{2|\mathcal{S}_x|} \displaystyle\sum_{i \in \mathcal{S}_x} d(s, \tau^0_i) - \dfrac{1}{2|\mathcal{F}_x|} \displaystyle\sum_{i \in \mathcal{F}_x} d(s, \tau^0_i), & \text{otherwise.} \end{cases} \tag{8}$$

For an all-success group, utility increases when the skill raises the likelihood of verified successful behavior. For an all-failure group, the sign is reversed, so utility increases when the skill suppresses observed failed behavior. For a mixed group, the success-support and failure-suppression terms receive equal total weight regardless of their relative counts. We refer to this balanced quantity as the *group-aware skill utility*.

**Anchor-based utility.** The selected successful Scout supplies the scoring anchor as well as the generation evidence:

$$U^{\text{anc}}(s; x) = d(s, \tau^0_{a_x}), \qquad \mathcal{S}_x \neq \varnothing. \tag{9}$$

Anchor-based utility uses one successful Scout, whereas group-based utility also incorporates failure evidence.

Skill utility provides a policy-relative signal for candidate ranking and abstention. By comparing outcome-conditioned likelihoods over verified Scout trajectories, it identifies skills aligned with the current Executor, while environment outcomes supervise Executor learning. Appendix C provides a log-likelihood-ratio interpretation and corresponding selection guarantees.

**Positive-gated selection.** Let $U_j = U^v(s_j; x)$, and let $\mathcal{V}_x$ contain candidates that satisfy the required format and have a finite utility. If $\mathcal{V}_x$ is nonempty, we define $j^* = \arg\max_{j \in \mathcal{V}_x} U_j$ and select

$$s^*_x = \begin{cases} s_{j^*}, & \mathcal{V}_x \neq \varnothing \text{ and } U_{j^*} > 0, \\ \varnothing, & \text{otherwise.} \end{cases} \tag{10}$$

The ranking identifies the most compatible candidate, while the sign test determines whether any skill should be injected.

### 4.4 Adaptive Skill Injection Rollout

Given the selected context $s^*_x$, the Executor samples $L$ fresh *Post-Scout* trajectories, $\tau^1_k \sim \pi^{\text{exec}}_\theta(\cdot \mid x, s^*_x; \mathcal{T})$, $k = 1, \ldots, L$. Thus, every query proceeds through the same Executor interface: a positive-utility skill is supplied when available, and $s^*_x = \varnothing$ otherwise.

### 4.5 Joint Policy Optimization

**Role-specific learning signals.** The $N$ Scouts and $L$ post-selection trajectories form one Executor GRPO group whose rewards come exclusively from the environment verifier. The $M$ generated skill texts form a separate Generator GRPO group. Their detached rewards are derived from skill utility:

$$r^{\text{gen}}_j = \begin{cases} \max\{0, \text{sg}[U_j]\}, & v = \text{anc}, \\ \text{sg}[U_j], & v = \text{grp}, \end{cases} \tag{11}$$

Because anchor evidence contains only successes, nonpositive utilities are clipped to zero; group-based rewards retain their signs to distinguish success-supporting from failure-reinforcing candidates. Executor and Generator rewards are normalized separately.

We optimize the shared parameters with

$$J_{\text{SS}}(\theta) = J^{\text{exec}}_{\text{GRPO}}(\theta; R) + \lambda_{\text{gen}} J^{\text{gen}}_{\text{GRPO}}(\theta; r^{\text{gen}}) + \eta_{\text{gen}} \mathcal{H}_{\text{gen}}(\theta), \tag{12}$$

where $\lambda_{\text{gen}}$ weights Generator learning and $\eta_{\text{gen}}$ controls its entropy regularization. The Executor learns from verified task outcomes, while the Generator learns from policy-relative utility. Because they share $\theta$, both skill-free Scouts and selectively scaffolded Post-Scouts update the same policy, reinforcing useful scaffolded behavior without separate distillation.

**Skill-free deployment.** At deployment, we disable the skill pathway and use $\pi^{\text{exec}}_{\theta^*}(\cdot \mid x, \varnothing, h_t; \mathcal{T})$, eliminating Scout collection and all skill-related inference overhead.

**Table 1. Performance on WebShop and ALFWorld with Qwen3-8B.** We report success rates (%) across task categories and their averages. † denotes evaluation with skills, while ∗ denotes evaluation without skills. **Best** and second-best results are marked accordingly.

| | WebShop | | | | | | | | | | ALFWorld | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ID | | | | | OOD | | | | Avg. | Task Type | | | | | | Avg. |
| Method | App. | Ele. | Foo. | Oth. | ID | Acc. | Bea. | Hom. | OOD | | Pic. | Hea. | Coo. | Cle. | Loo. | Pk2. | |
| ReAct | 18.8 | 16.9 | 18.1 | 17.0 | 17.6 | 14.1 | 16.0 | 8.6 | 12.6 | 16.1 | 59.4 | 32.1 | 41.1 | 29.8 | 41.0 | 5.2 | 35.6 |
| *Prompt-based Agentic / Memory-based Methods* | | | | | | | | | | | | | | | | | |
| Reflexion | 22.7 | 19.5 | 20.8 | 21.1 | 20.9 | 16.7 | 24.5 | 13.0 | 17.5 | 19.8 | 59.9 | 21.7 | 42.9 | 41.5 | 38.9 | 25.7 | 39.3 |
| Mem0 | 18.3 | 15.7 | 14.5 | 17.0 | 16.4 | 13.3 | 15.3 | 8.6 | 12.1 | 15.1 | 37.0 | 25.0 | 41.7 | 41.9 | 30.6 | 29.4 | 35.0 |
| ExpeL | 25.2 | 22.5 | 26.5 | 24.3 | 24.4 | 17.3 | 20.4 | 9.6 | 15.4 | 21.6 | 32.3 | 48.9 | 47.6 | 37.5 | 41.7 | 29.4 | 39.7 |
| MemP | 23.8 | 18.3 | 22.1 | 18.4 | 20.4 | 13.6 | 13.9 | 9.9 | 12.4 | 17.9 | 53.1 | 46.2 | 44.1 | 55.2 | 29.2 | 32.4 | 45.2 |
| w/ Skill | 12.5 | 13.2 | 17.7 | 14.9 | 14.3 | 11.1 | 18.5 | 6.3 | 11.4 | 13.4 | 56.3 | 8.2 | 43.5 | 45.6 | 43.1 | 16.9 | 36.8 |
| *RL Methods* | | | | | | | | | | | | | | | | | |
| GRPO | 36.2 | 36.4 | 36.8 | 44.6 | 38.2 | 29.2 | 43.8 | 28.8 | 32.9 | 36.5 | 97.4 | 86.4 | 92.3 | 82.3 | 71.5 | 80.9 | 85.6 |
| GiGPO | 33.7 | 46.6 | 28.5 | 52.8 | 41.2 | **39.4** | 54.4 | 30.0 | 40.0 | 40.8 | 87.0 | 82.1 | 92.3 | 75.8 | 61.1 | 53.7 | 76.7 |
| *Skill-based RL Methods* | | | | | | | | | | | | | | | | | |
| INSPO† | 33.6 | 40.9 | 34.7 | 47.4 | 39.3 | 30.8 | 46.3 | 30.8 | 34.8 | 37.9 | 82.3 | 73.4 | 91.1 | 70.2 | 62.5 | 58.8 | 73.7 |
| SkillRL† | 31.9 | 37.3 | 38.6 | 50.3 | 39.1 | 36.7 | 46.5 | 31.5 | 37.4 | 38.6 | 84.7 | 82.8 | 90.2 | 95.8 | **98.6** | 96.8 | 91.2 |
| SKILL1† | 33.5 | 36.6 | 28.0 | 48.3 | 36.7 | 29.7 | 53.9 | 28.4 | 35.6 | 36.4 | 91.7 | **95.7** | 90.5 | 83.9 | 94.4 | **100.0** | 91.8 |
| Evolving-RL† | **41.7** | 41.1 | 36.4 | 52.0 | 42.7 | 36.1 | **55.8** | 31.7 | 39.7 | 41.8 | 92.2 | 71.2 | 76.2 | 61.3 | 66.0 | 87.5 | 74.8 |
| SEED∗ | 34.4 | 41.9 | 30.6 | 53.4 | 40.3 | 36.6 | 52.1 | 33.2 | 39.4 | 40.1 | 84.4 | 83.7 | 89.3 | 92.7 | 83.3 | 77.2 | 85.9 |
| Evolving-RL∗ | 26.2 | 31.7 | 25.0 | 29.8 | 28.6 | 21.7 | 39.4 | 20.2 | 25.8 | 27.7 | 99.0 | 70.1 | 79.2 | 57.7 | 85.4 | 78.7 | 77.0 |
| **SkillScaffold**∗ | 38.3 | **47.6** | **42.7** | **57.9** | **46.6** | 37.7 | **55.8** | **36.5** | **42.0** | **45.1** | **100.0** | 87.0 | **100.0** | **96.8** | 88.9 | 94.1 | **94.8** |

# 5 Experiments

## 5.1 Experimental Setup

**Benchmarks.** We evaluate SkillScaffold on four interactive benchmarks. **ALFWorld** (Shridhar et al., 2021) is a text-based household environment requiring multi-step planning and object interaction. **WebShop** (Yao et al., 2022a) is an online-shopping simulator in which agents search for products matching user specifications. **ScienceWorld** (Wang et al., 2022) evaluates interactive scientific reasoning and experimentation. These three are short-horizon settings. In contrast, $\tau^2$**-Bench** (Yao et al., 2024) evaluates long-horizon tool-agent-user interaction across Airline, Retail, and Telecom. We report success rate for the first three benchmarks and Pass@1 for $\tau^2$-Bench. Dataset and evaluation details are provided in Appendix A.1.

**Baselines and Implementation Details.** All main runs use Qwen3-8B. Baselines include (1) **Prompt- and memory-based methods:** ReAct (Yao et al., 2022b), Reflexion (Shinn et al., 2023), Mem0 (Chhikara et al., 2025), ExpeL (Zhao et al., 2024), MemP (Fang et al., 2026), and direct skill prompting. (2) **RL methods:** GRPO (Shao et al., 2024) and GiGPO (Feng et al., 2025) optimize the policy without skill augmentation. (3) **Skill-based RL methods:** INSPO (Zhou et al., 2026) and SkillRL (Xia et al., 2026) use teacher-generated guidance, whereas Skill1 (Shi et al., 2026), Evolving-RL (Fan et al., 2026), and SEED (Wu et al., 2026) generate or distill guidance from the policy's own experience. Full implementation details are provided in Appendix A.

## 5.2 Main Results

**Overall performance.** Tables 1 and 2 show a clear progression over increasingly competitive baselines. First, SkillScaffold substantially outperforms prompt- and memory-based methods, with gains of 23.5 points on WebShop and 49.6 on ALFWorld over their strongest results. It then surpasses standard RL methods across all four benchmarks, including gains of 9.2 points on ALFWorld and 4.2 on ScienceWorld. Finally, SkillScaffold remains best against skill-based RL methods on every benchmark, improving the strongest such results by 3.3 points on WebShop and 3.0 on ALFWorld. Its 6.3-point gain on Telecom further extends this advantage to long-horizon tool use. Full training dynamics are provided in Appendix B.

**Table 2. Performance on ScienceWorld and $\tau^2$-Bench with Qwen3-8B.** We report success rates (%) on the ScienceWorld ID/OOD splits and the three $\tau^2$-Bench domains. Teal arrows indicate absolute gains in percentage points over the strongest competing result. **Best** and second-best results are marked accordingly.

| Method | ScienceWorld | | | $\tau^2$-Bench | | | |
|---|---|---|---|---|---|---|---|
| | ID | OOD | Avg. | Airline | Retail | Telecom | Avg. |
| ReAct | 19.3 | 11.1 | 16.0 | 17.8 | 35.5 | 7.6 | 20.3 |
| SFT | – | – | – | 41.0 | 40.1 | 56.6 | 45.9 |
| GRPO | 92.6 | 43.6 | 72.9 | 45.8 | 57.2 | 77.0 | 60.0 |
| SkillRL | 88.2 | 47.7 | 71.9 | **46.0** | 57.9 | 75.7 | 59.9 |
| **SkillScaffold** | **96.2** ↑3.6 | **48.8** ↑1.1 | **77.1** ↑4.2 | **46.0** | **60.8** ↑2.9 | **83.3** ↑6.3 | **63.4** ↑3.4 |

**Skill-free deployment and generalization.** SkillScaffold removes the Generator and all textual skills at inference, yet outperforms skill-conditioned baselines on both WebShop and ALFWorld. On WebShop, it reaches 45.1 without skills, compared with 41.8 for Evolving-RL with skills and 27.7 after its skills are removed. The gains also hold across ID and OOD splits: SkillScaffold reaches 46.6/42.0 on WebShop and 96.2/48.8 on ScienceWorld, exceeding the strongest competing results on both splits. Thus, the improvements do not rely on retaining an external skill pathway and generalize beyond the training categories. Full Pass@$k$ results are provided in Appendix Table 6.

### 5.3 Ablation Studies

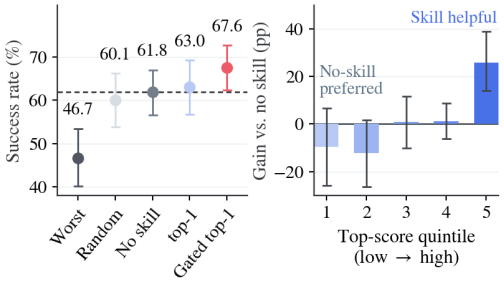
Table 3 isolates utility-supervised Generator learning, utility-based candidate selection, and positive-gated skill injection. Removing Generator learning reduces ALFWorld/WebShop performance to 89.6/39.5, showing that skill generation must adapt to the evolving policy. Random selection and shuffled candidate–utility assignments yield 79.9/39.2 and 82.8/37.1, showing that utility must both guide selection and remain paired with the candidate it evaluates. Removing the positive gate gives 85.8/39.6, confirming that even the top-ranked skill can be harmful when its utility is nonpositive. Finally, training the Generator without injecting selected skills into Executor rollouts gives 90.3/41.5, showing that Generator supervision alone cannot replace skill-scaffolded exploration.

**Table 3. Ablation results on ALFWorld and WebShop.**

| Variant | ALF. | Web. |
|---|---|---|
| **SkillScaffold**(Full) | **94.8** | **45.1** |
| *Generator Learning* | | |
| w/o Generator learning | 89.6 | 39.5 |
| No Gen. loss + random selection | 81.3 | 40.3 |
| *Utility Selection* | | |
| Random selection | 79.9 | 39.2 |
| Shuffled utility | 82.8 | 37.1 |
| *Selective Scaffolding* | | |
| w/o abstention | 85.8 | 39.6 |
| No Executor injection | 90.3 | 41.5 |

### 5.4 Validating Skill Utility for Credit Assignment

We reuse the independently evaluated query–candidate sets from the pilot study. For each candidate, the likelihood-based utility is computed only from the fixed Scout trajectories, without using its downstream evaluation rollouts. Figure 5(a) applies four decision rules to the same candidate sets: skill-free execution, random injection, forced injection of the utility-ranked top-1 candidate, and top-1 injection only when its utility is positive. Their success rates are 61.8%, 60.1%, 63.0%, and 67.6%, respectively. This shows that ranking alone provides only a modest gain, whereas positive gating is critical for avoiding unhelpful or harmful injections. Panel (b) partitions candidate–query pairs into five utility quintiles and uses independent downstream rollouts only for evaluation, reporting gain over skill-free execution. The measured gain rises with utility, supporting its use for both ranking and abstention.




**(a) Selective skill injection (b) Top score predicts utility**

**Figure 5. Skill-utility validation.**

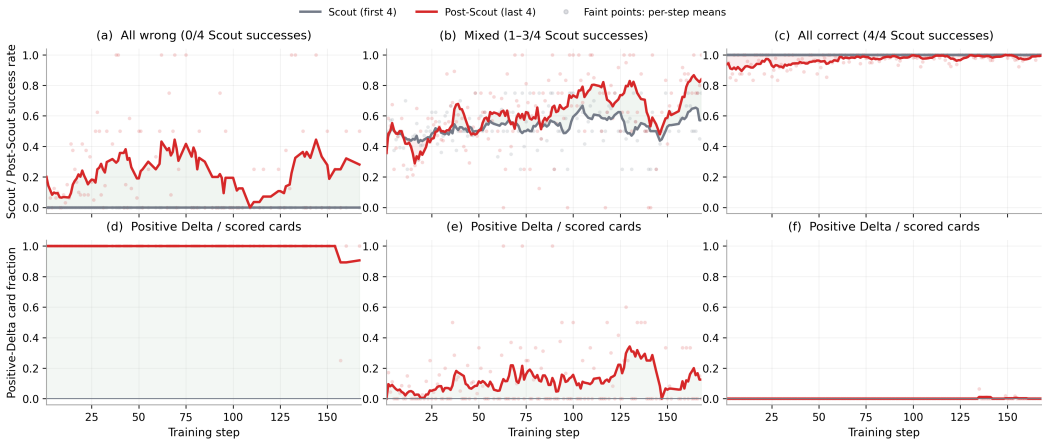



**Figure 6. ALFWorld exploration dynamics.** Scout/Post-Scout success (top) and skill-injection rates (bottom), grouped by all-wrong, mixed, and all-correct Scout outcomes.

### 5.5 SkillScaffold Improves Training-Time Exploration

Figure 6 shows that scaffolding concentrates on queries where skill-free exploration is weakest. Relative to Scouts, Post-Scout success improves by 21.1 points for all-wrong groups. This indicates that selected skills can unlock successful exploration even when skill-free Scouts discover no solution. The gain is 5.4 points for mixed groups, while performance changes little for all-correct groups ($-2.3$ points). Correspondingly, the gate injects skills almost universally for all-wrong groups, selectively for mixed groups, and rarely for all-correct groups. Cross-benchmark curves are reported in Appendix B.

### 5.6 Skill Internalization into Model Parameters

Table 4 tests whether strategies first exposed through selected skills become encoded in the shared model parameters. We fix cohorts of training queries that benefit from skill-assisted exploration at an early checkpoint and reevaluate them at later checkpoints with the skill pathway disabled. Skill-free success improves by 19.7 points on WebShop, 29.6 on ALFWorld, 65.0 on ScienceWorld, and 16.6 on $\tau^2$-Bench; intervals exclude zero on the latter three benchmarks. Because the later Executor receives no skill text, these gains indicate that shared-parameter training transfers skill-guided strategies into the policy itself. A qualitative internalization case is provided in Appendix D.1.

**Table 4. Skill-free evidence of internalization.** Later-minus-early skill-free success on fixed early skill-assisted training cohorts, with 95% bootstrap intervals.

| Benchmark | $N$ | Recovery (pp) |
|---|---|---|
| WebShop | 19 | +19.7 [0.0, 39.5] |
| ALFWorld | 54 | +29.6 [20.4, 38.9] |
| ScienceWorld | 50 | +65.0 [56.0, 73.5] |
| $\tau^2$-Bench | 98 | +16.6 [10.7, 22.5] |

## 6 Conclusion

We introduced SkillScaffold, a policy-adaptive training scaffold for skill credit assignment in agentic RL. Rather than assuming generated skills are helpful or evaluating every candidate through additional environment interaction, SkillScaffold reuses outcome-verified trajectories to estimate policy-relative utility through outcome-aware likelihood comparisons. This signal supports candidate ranking, abstention, and Generator learning, while environment outcomes continue to supervise the Executor. Across diverse short- and long-horizon benchmarks, SkillScaffold improves over strong agentic-RL and skill-based RL baselines under fully skill-free inference. Overall, SkillScaffold selectively uses skills to improve training-time exploration, absorbs their guidance into the policy, and removes the skill pathway at deployment.

### Reproducibility Statement

Section 4 describes SkillScaffold and its training objective. Appendices A–D provide implementation details, additional results, and complete prompts and examples. We will provide the implementation code in the supplementary material.

### AI Use Statement

Generative AI tools were used to assist with language editing, LaTeX formatting, and code navigation. They were not treated as sources of evidence or used to autonomously determine the experimental design, analysis, or conclusions. The authors reviewed and verified all AI-assisted text and code and take full responsibility for the scientific claims and final manuscript.

## References

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. Automanual: Constructing instruction manuals by llm agents via interactive environmental learning. *Advances in Neural Information Processing Systems*, 37:589–631, 2024.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

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.

Guanting Dong et al. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*, 2025.

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

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, 2026.

Lang Feng et al. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*, 2025.

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

Jiaxuan Gao, Jiaao Chen, Chuyi He, Shusheng Xu, Di Jin, and Yi Wu. From self-evolving synthetic data to verifiable-reward rl: Post-training multi-turn interactive tool-using agents. *arXiv preprint arXiv:2601.22607*, 2026.

Zelin He, Haotian Lin, Boran Han, Wei Zhu, Haoyang Fang, Bernie Wang, Xuan Zhu, Runze Li, and Matthew Reimherr. Reskill: Reconciling skill creation with policy optimization in agentic rl. *arXiv preprint arXiv:2606.01619*, 2026.

Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. SKILL0: In-context agentic reinforcement learning for skill internalization. *arXiv preprint arXiv:2604.02268*, 2026.

Kolby Nottingham, Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Sameer Singh, Peter Clark, and Roy Fox. Skill set optimization: Reinforcing language model behavior via transferable skills. *arXiv preprint arXiv:2402.03244*, 2024.

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

Junhao Shen, Teng Zhang, Xiaoyan Zhao, and Hong Cheng. Dynamic skill lifecycle management for agentic reinforcement learning. *arXiv preprint arXiv:2605.10923*, 2026.

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

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

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, and Honggang Qi. Skill-SD: Skill-conditioned self-distillation for multi-turn llm agents. *arXiv preprint arXiv:2604.10674*, 2026a.

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

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

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

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. Seed: Self-evolving on-policy distillation for agentic reinforcement learning. *arXiv preprint arXiv:2607.14777*, 2026.

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

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

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. $\tau$-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Zhiwei Zhang, Yudi Lin, Nikki Lijing Kuang, Linlin Wu, Xiaomin Li, Songtao Liu, and Fenglong Ma. Co-evolving skill generation and policy optimization. *arXiv preprint arXiv:2606.08755*, 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.

Han Zhou, Xingchen Wan, Ivan Vulić, and Anna Korhonen. Agentic policy optimization via instruction-policy co-evolution. *arXiv preprint arXiv:2512.01945*, 2026.

Jiapeng Zhu, Jianxiang Yu, Yibo Zhao, Chengcheng Han, Qi Gu, Xunliang Cai, Xiang Li, and Weining Qian. Skill0.5: Joint skill internalization and utilization for out-of-distribution generalization in agentic reinforcement learning. *arXiv preprint arXiv:2605.28424*, 2026.

# A Implementation Details

## A.1 Benchmark and Data Details

**ALFWorld.** ALFWorld contains six household task categories: Pick, Heat, Cool, Clean, Look, and Pick2. We train on the full 3,553-task split, comprising 790 Pick, 459 Heat, 533 Cool, 650 Clean, 308 Look, and 813 Pick2 tasks, and evaluate success rate on the 134-task valid-unseen split.

**WebShop.** Following the category-generalization protocol of Skill0.5 (Zhu et al., 2026), we partition the 12,087 human-annotated WebShop goals into seven domains using keyword-based classification of goal instructions. Apparel, Electronics, Footwear, and Other are used for training and ID evaluation; Accessories, Beauty & Health, and Home Decor are held out for OOD evaluation. Because Other initially accounts for over 60% of the goals, we apply farthest-point sampling over sentence embeddings to balance its size.

The resulting training split contains 3,320 ID goals: 776 Apparel, 938 Electronics, 606 Footwear, and 1,000 Other goals, with Other downsampled from approximately 6,600. The ID validation split contains 454 goals: 113 Apparel, 152 Electronics, 89 Footwear, and 100 Other, with Other downsampled from approximately 940. The OOD validation split contains 207 goals: 80 Accessories, 54 Beauty & Health, and 73 Home Decor. Training uses original goal indices 1,500 onward, while the validation pool combines the original test indices 0–499 and development indices 500–1,499.

**ScienceWorld.** Following ReSkill (He et al., 2026), we use the electricity split, which contains two task types: identifying power components and testing conductivity. The action space contains approximately 26 verb templates, and the interaction horizon is at most 50 steps. Training uses 460 variations: 10 `power-component` and 450 `test-conductivity` variations under the `easy` simplification. Test-ID uses 230 development episodes with held-out variations of the same task types; Test-OOD uses 155 episodes from related variants involving unknown-material conductivity and renewable energy. We report ID and OOD success rates.

**$\tau^2$-Bench.** $\tau^2$-Bench evaluates long-horizon dialogue, tool execution, and policy compliance in Airline, Retail, and Telecom. Unlike the other three benchmarks, all RL-based baselines and SkillScaffold are trained from the same SFT checkpoint. The user role is simulated by DeepSeek-V4-Flash. Both the SFT and RL data are derived from the synthetic data released by AReaL-SEA (Gao et al., 2026). We estimate task pass rates with the SFT model and remove all-success and all-failure groups, which provide no group-relative learning signal. The resulting RL pool contains 300 entries per domain, or 900 in total.

## A.2 Benchmark Configurations

Table 5 summarizes the configurations used during training. ALFWorld, WebShop, and ScienceWorld are initialized from the base Qwen3-8B model, whereas all RL-based methods on $\tau^2$-Bench use the same domain-adapted SFT checkpoint.

**Table 5.** Training configurations for the reported SkillScaffold models.

| Parameter | ALFWorld | WebShop | ScienceWorld | $\tau^2$-Bench |
|---|---|---|---|---|
| *Reinforcement learning* | | | | |
| Model family | Qwen3-8B | Qwen3-8B | Qwen3-8B | Qwen3-8B |
| Training steps | 100 | 200 | 160 | 200 |
| Optimizer | Adam | Adam | Adam | Adam |
| Learning rate (constant) | $10^{-6}$ | $10^{-6}$ | $10^{-6}$ | $10^{-6}$ |
| Adam $(\beta_1, \beta_2)$ | $(0.9, 0.98)$ | $(0.9, 0.98)$ | $(0.9, 0.98)$ | $(0.9, 0.999)$ |
| Query batch size | 16 | 16 | 16 | 32 |
| Configured global batch size | 128 | 128 | 128 | 32 |
| Executor group size | 8 | 8 | 8 | 8 |
| PPO clip $\epsilon$ (lower / upper) | 0.20 / 0.28 | 0.20 / 0.28 | 0.20 / 0.28 | 0.20 / 0.28 |
| KL penalty | 0 | 0 | 0 | 0 |
| Executor entropy coefficient | 0 | 0 | 0 | 0 |
| Executor temperature | 1.0 | 1.0 | 1.0 | 0.7 |
| Executor top-$p$ | 0.85 | 0.85 | 0.85 | 1.0 |
| Maximum interaction steps | 50 | 15 | 50 | 160 |
| Maximum Executor tokens / turn | 512 | 512 | 384 | 1,024 |
| Executor sequence token budget | 32,768 | 16,384 | 32,768 | 32,768 |
| *Skill generation and selection* | | | | |
| Utility variant | Group-based | Anchor-based | Group-based | Group-based |
| $N$ Scouts / $M$ candidates / $L$ Post-Scouts | 4 / 4 / 4 | 4 / 4 / 4 | 4 / 4 / 4 | 4 / 4 / 4 |
| Maximum Generator tokens | 900 | 256 | 900 | 700 |
| Generator-loss weight $\lambda_{\text{gen}}$ | 0.1 | 0.1 | 0.1 | 0.1 |
| Generator entropy coefficient | 0.002 | 0 | 0.002 | 0.002 |
| Injection threshold | $U > 0$ | $U > 0$ | $U > 0$ | $U > 0$ |
| Test-time external skill | None | None | None | None |

### A.3 End-to-End Training Procedure

Algorithm 1 summarizes one complete SkillScaffold training update. Environment rewards supervise the Executor, whereas detached policy-relative utility supervises the Generator. Both objectives update the same policy parameters.

### A.4 Evidence Construction and Candidate Generation

For group-based training, the evidence record contains the task, all $N$ Scout trajectories, and their verifier outcomes. All-success groups emphasize reusable successful decisions, all-failure groups request conditional corrections without inventing an unobserved solution, and mixed groups contrast successful and failed decisions. For anchor-based training, the first verified successful Scout in sampling order is used as the evidence anchor. If no Scout succeeds, candidate generation and Generator optimization are skipped for that query, and Executor execution remains skill-free.

Each evidence record produces $M$ independently sampled candidate skills. Candidates must satisfy the prescribed card format and express reusable decision guidance rather than copy task-specific answers. Formatting validity is checked separately from utility availability: an invalid candidate is never selected, while an incomplete Generator group is omitted according to the rules in Section 4.5.

### A.5 Diagnostic Evaluation Protocols

**Skill-utility validation.** The comparison in Section 5.4 applies every injection rule to the same fixed query–candidate sets. Candidate utilities are computed from the fixed Scout trajectories, while separate downstream rollouts are reserved for evaluation. The utility-quintile analysis groups candidate–query pairs by the estimated utility and reports their independently measured gain over skill-free execution.

**Algorithm 1** SkillScaffold training with policy-relative skill utility

**Require:** Policy $\pi_\theta$, task distribution $\mathcal{D}$, tool engine $\mathcal{T}$, utility variant $v \in \{\text{grp}, \text{anc}\}$
**Require:** $N$ Scouts, $M$ candidate skills, $L$ Post-Scout rollouts
**for** each training update **do**
  Sample $x \sim \mathcal{D}$
  Collect skill-free Scouts $\{(\tau_i^0, y_i)\}_{i=1}^N$ with $\pi_\theta^{\text{exec}}$
  Construct evidence $E_x^v$ from Scout trajectories and outcomes
  **if** $v$ = anc and no Scout succeeds **then**
    Set $C_x \leftarrow \varnothing$ and $s_x^* \leftarrow \varnothing$
  **else**
    Generate $C_x = \{s_j\}_{j=1}^M$ using $\pi_\theta^{\text{gen}}(\cdot \mid E_x^v)$
    **for** each format-valid candidate $s_j$ **do**
      Score the fixed Scout tokens under $s_j$ and the skill-free context
      Compute utility $U_j \leftarrow U^v(s_j; x)$
    **end for**
    Let $\mathcal{V}_x$ contain candidates with valid format and finite utility
    **if** $\mathcal{V}_x = \varnothing$ **then**
      $s_x^* \leftarrow \varnothing$
    **else**
      $j^* \leftarrow \arg\max_{j \in \mathcal{V}_x} U_j$
      $s_x^* \leftarrow s_{j^*}$ if $U_{j^*} > 0$; otherwise $s_x^* \leftarrow \varnothing$
    **end if**
  **end if**
  Collect $L$ fresh Post-Scout trajectories conditioned on $s_x^*$
  Form the Executor GRPO group from the $N + L$ trajectories using environment rewards
  Form Generator rewards using Equation 11 when a complete trainable group is available
  Update shared parameters with $J_{\text{GRPO}}^{\text{exec}} + \lambda_{\text{gen}} J_{\text{GRPO}}^{\text{gen}} + \eta_{\text{gen}} \mathcal{H}_{\text{gen}}$; omit Generator terms when unavailable
**end for**
**return** skill-free policy $\pi_{\theta^*}^{\text{exec}}(\cdot \mid x, \varnothing; \mathcal{T})$

**Skill-free strategy acquisition.** For each benchmark, we form a fixed cohort at an early checkpoint from training queries that benefit from skill-assisted exploration. Cohort membership is frozen before later evaluation. The same queries are then reevaluated at a later checkpoint with the skill pathway disabled, and recovery is computed as the later-minus-early skill-free success rate. Confidence intervals use a query-level bootstrap.

# B Additional Experimental Results

## B.1 Full $\tau^2$-Bench Pass@$k$ Results

Table 6 expands the main Pass@1 comparison with Pass@4 and Pass@8 for every domain. The average columns are macro averages over Airline, Retail, and Telecom.

**Table 6.** Full $\tau^2$-Bench Pass@$k$ results (%) with Qwen3-8B. Average columns are macro averages over Airline, Retail, and Telecom. Teal arrows indicate absolute gains in percentage points over the strongest competing result.

| | Airline | | | Retail | | | Telecom | | | Average | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | P@1 | P@4 | P@8 | P@1 | P@4 | P@8 | P@1 | P@4 | P@8 | P@1 | P@4 | P@8 |
| ReAct | 17.8 | 34.0 | 46.0 | 35.5 | 61.7 | 70.2 | 7.6 | 23.1 | 32.5 | 20.3 | 39.6 | 49.5 |
| SFT | 41.0 | 62.1 | 74.0 | 40.1 | 71.3 | 84.2 | 56.6 | 91.8 | 97.4 | 45.9 | 75.1 | 85.2 |
| GRPO | 45.8 | 63.8 | 70.0 | 57.2 | 85.4 | **92.1** | 77.0 | 97.2 | 99.1 | 60.0 | 82.1 | 87.1 |
| SkillRL | **46.0** | 64.2 | 70.0 | 57.9 | 83.6 | 90.4 | 75.7 | 95.3 | 99.1 | 59.9 | 81.0 | 86.5 |
| **SkillScaffold** | **46.0** | **70.3** ↑6.1 | **78.0** ↑4.0 | **60.8** ↑2.9 | **85.6** ↑0.2 | 91.2 | **83.3** ↑6.3 | **98.5** ↑1.3 | **100.0** ↑0.9 | **63.4** ↑3.4 | **84.8** ↑2.7 | **89.7** ↑2.6 |

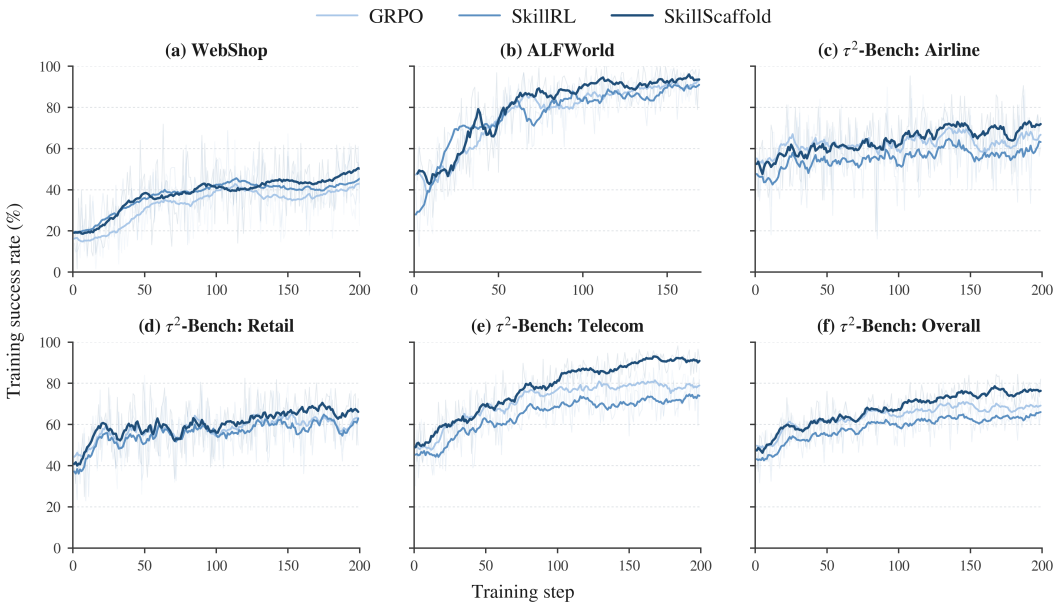
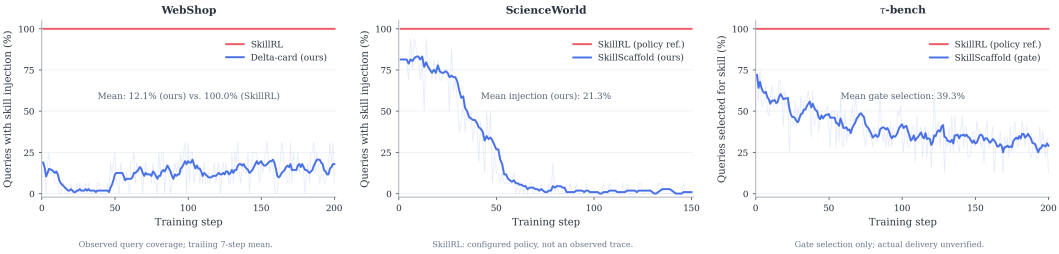



**Figure 7. Training success-rate curves across benchmarks.** GRPO, SkillRL, and SkillScaffold on (a) WebShop, (b) ALFWorld, (c) $\tau^2$-Bench Airline, (d) Retail, (e) Telecom, and (f) the $\tau^2$-Bench macro average. Dark curves show smoothed trends and faint curves show raw measurements.




(a) WebShop (b) ScienceWorld (c) $\tau^2$-Bench

**Figure 8. Training-time skill injection and selection rates.** Panels (a)–(b) report skill injection on WebShop and ScienceWorld; panel (c) reports positive-utility selection on $\tau^2$-Bench.

### B.2 Training Dynamics Across Benchmarks

Figure 7 reports training curves for WebShop, ALFWorld, and the three individual $\tau^2$-Bench domains. The largest late-training separation appears on Telecom, and the Airline and Retail curves also retain advantages at later checkpoints.

### B.3 Skill Injection Rates Across Benchmarks

Figure 8 reports how frequently the training-time scaffold is activated beyond ALFWorld. The always-on SkillRL reference supplies a skill to every query, whereas SkillScaffold invokes its scaffold selectively. The mean observed injection rate is 12.1% on WebShop and 21.3% on ScienceWorld. On $\tau^2$-Bench, the positive-utility gate selects a skill for 39.3% of queries on average.

### B.4 Cross-Benchmark Post-Scout Exploration Dynamics

Figures 9–12 test whether the selective-scaffolding behavior observed on ALFWorld also generalizes to the other benchmarks. Queries are grouped by the outcomes of four skill-free Scouts. The upper row compares Scout and fresh Post-Scout rewards, while the lower row reports the positive-utility candidate rate.

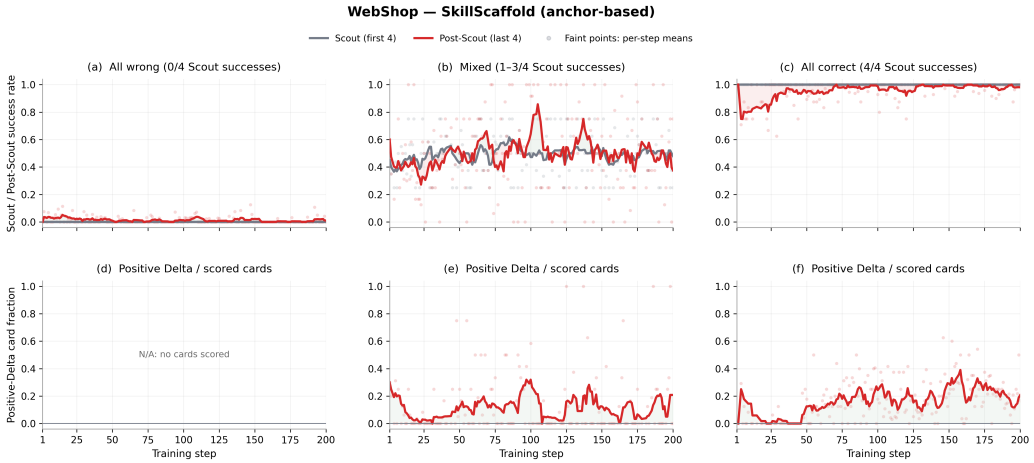



**Figure 9. Selective anchor-guided exploration on WebShop.** Using a verified successful Scout as evidence, the anchor-based variant selectively guides mixed groups and maintains strong Post-Scout performance as the Scout policy approaches the all-correct regime.

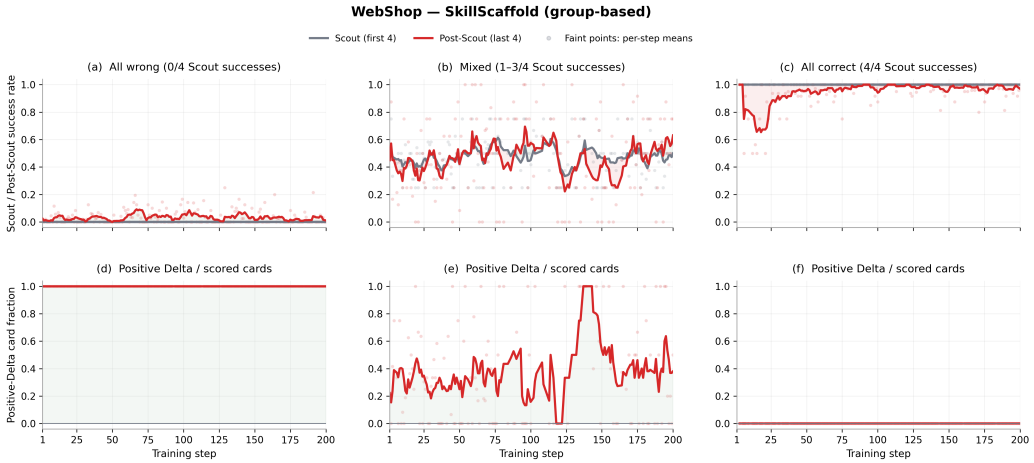



**Figure 10. Corrective group-guided exploration on WebShop.** Outcome-labeled Scout groups support corrective skill generation and produce persistent nonzero Post-Scout success for initially all-wrong queries. Skill activation concentrates on all-wrong and mixed groups and naturally diminishes once all Scouts succeed.

Across benchmarks, SkillScaffold concentrates positive-utility skills on all-wrong and mixed groups, where Post-Scout rollouts recover additional successful behavior. Skill use declines as the Scout policy becomes competent and is rare once all Scouts succeed, focusing external guidance on queries where it improves exploration.

## C Theoretical Analysis of Skill Utility for Credit Assignment

This section analyzes the proposed likelihood-based skill utility as a policy-relative surrogate. We derive its log-likelihood-ratio interpretation, ranking and abstention robustness, a calibrated downstream-return bound, and its interaction and computation complexity.

### C.1 Log-Likelihood-Ratio Interpretation

For the Executor-token index set $\mathcal{I}(\tau)$, define the conditional trajectory likelihood

$$p_\theta(\tau \mid x, c) = \prod_{t \in \mathcal{I}(\tau)} \pi_\theta^{\text{exec}}(u_t \mid x, c, h_t^\tau). \tag{13}$$

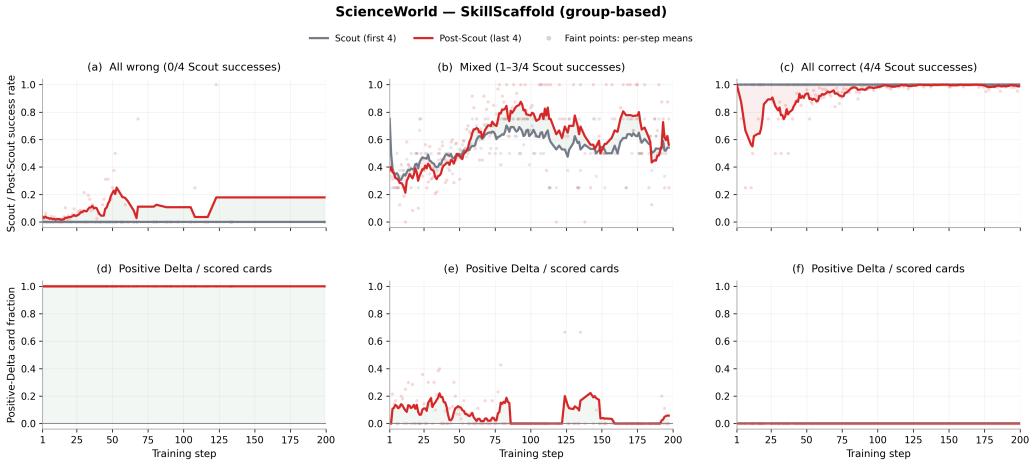



**Figure 11. Competence-adaptive exploration on ScienceWorld.** Group-based scaffolding creates successful trajectories for initially all-wrong groups and improves mixed-group performance during later training. Positive-utility candidates become less frequent as the Scout policy improves, reflecting competence-adaptive skill use.

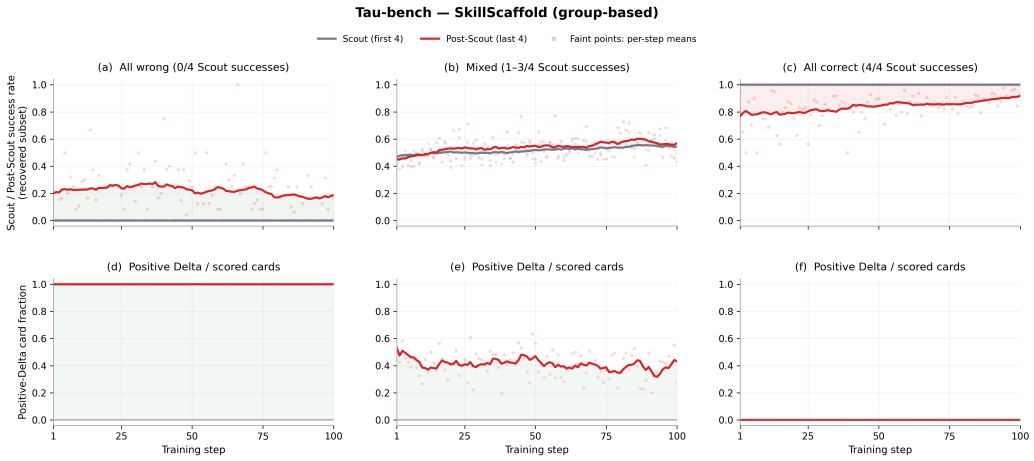



**Figure 12. Competence-adaptive exploration on $\tau^2$-Bench.** Post-Scout rollouts recover successful trajectories from initially all-wrong groups and improve performance across much of the mixed regime. Positive-utility candidates are frequent for all-wrong groups, selective for mixed groups, and unnecessary once all Scouts succeed.

Equation 7 then satisfies the exact identity

$$d(s,\tau) = \frac{1}{|I(\tau)|} \log \frac{p_\theta(\tau \mid x, s)}{p_\theta(\tau \mid x, \varnothing)}. \tag{14}$$

A positive $d(s,\tau)$ means that conditioning on $s$ increases the geometric mean token likelihood assigned to the observed Executor behavior.

**Proposition 1 (balanced signed evidence).** For a mixed Scout group, group-based utility can be written as

$$U^{\mathrm{grp}}(s;x) = \frac{1}{2}\mathbb{E}_{i\sim \mathrm{Unif}(\mathcal{S}_x)}[d(s,\tau_i^0)] - \frac{1}{2}\mathbb{E}_{i\sim \mathrm{Unif}(\mathcal{F}_x)}[d(s,\tau_i^0)]. \tag{15}$$

Consequently, successful and failed evidence each receive total weight $1/2$, independent of their relative frequency in the Scout group.

*Proof.* Substituting the uniform expectations into Equation 15 gives the mixed branch of Equation 8. The coefficients sum to $1/2$ within each outcome class, so changing the class counts does not change their total weights. □

This balancing keeps successful and failed evidence equally influential, independent of their relative frequencies in the sampled Scout group.

### C.2 Selection Robustness under Utility Error

Let $U_j$ denote the population proxy utility of candidate $j$ under the current query and policy, and let $\widehat{U}_j$ be the utility estimated from the sampled Scout group. Assume a uniform error bound

$$|\widehat{U}_j - U_j| \le \varepsilon \qquad \text{for every eligible candidate } j. \tag{16}$$

**Proposition 2 (ranking and abstention bounds).** Let $\widehat{j} = \arg\max_j \widehat{U}_j$ and $j^* = \arg\max_j U_j$. Then

$$U_{\widehat{j}} \ge U_{j^*} - 2\varepsilon. \tag{17}$$

Moreover, if the positive gate selects $\widehat{j}$, then $U_{\widehat{j}} > -\varepsilon$; if the gate abstains, then $\max_j U_j \le \varepsilon$.

*Proof.* By optimality of $\widehat{j}$ under the estimated utilities,

$$U_{\widehat{j}} \ge \widehat{U}_{\widehat{j}} - \varepsilon \ge \widehat{U}_{j^*} - \varepsilon \ge U_{j^*} - 2\varepsilon. \tag{18}$$

If $\widehat{U}_{\widehat{j}} > 0$, the error bound gives $U_{\widehat{j}} > -\varepsilon$. If every estimated utility is nonpositive, then $U_j \le \widehat{U}_j + \varepsilon \le \varepsilon$ for every candidate. □

The proposition formalizes two roles of utility in skill credit assignment. Ranking incurs at most $2\varepsilon$ proxy regret, while abstention avoids overlooking any candidate whose proxy advantage exceeds the estimation-error scale.

### C.3 Connection to Downstream Return

Let

$$G(s; x) = \mathbb{E}[R(\tau; x) \mid x, s] - \mathbb{E}[R(\tau; x) \mid x, \varnothing] \tag{19}$$

denote the true downstream gain. The following local calibration condition connects policy-relative utility to this environment-level quantity.

**Assumption 1 (local surrogate calibration).** For eligible candidates under the current query and policy, suppose constants $a > 0$ and $b \ge 0$ exist such that

$$G(s_j; x) \ge aU_j - b. \tag{20}$$

**Proposition 3 (thresholded downstream bound).** Under Assumption 1 and Equation 16, selecting a candidate only when $\widehat{U}_j > t$ guarantees

$$G(s_j; x) > a(t - \varepsilon) - b. \tag{21}$$

In particular, $t \ge \varepsilon + b/a$ is sufficient for a nonnegative downstream-gain guarantee.

*Proof.* The estimation bound implies $U_j > t - \varepsilon$. Substitution into Equation 20 yields $G(s_j; x) > a(t - \varepsilon) - b$. □

The bound exposes a direct tradeoff between the threshold and the guaranteed downstream margin. Our experiments use $t = 0$, while the same gate supports a tunable positive margin when desired.

### C.4 Interaction and Computation Complexity

With $M$ candidates and $K$ evaluation episodes per condition, direct rollout utility estimation requires $(M + 1)K$ additional environment trajectories per query when the skill-free condition is shared. SkillScaffold launches no candidate-specific environment trajectories. It reuses $N$ Scouts already

collected for training and requires $MN$ candidate-conditioned trajectory scores plus $N$ cached skill-free scores, giving $O(MN)$ additional model scoring work. After selection, the $L$ post-selection trajectories are ordinary Executor training data rather than candidate-evaluation rollouts. With the default $N = M = L = 4$, each update uses eight environment trajectories in the Executor group and twenty reusable trajectory-context scores for utility computation.

# D Skill-Guided Execution Examples

The following pages provide the skill-generation prompt, the complete selected skill, and one skill-guided trajectory per benchmark. The ALFWorld and ScienceWorld prompts are abridged for space, while preserving their evidence rules, input structure, and required output format.

## D.1 Qualitative Case of Skill-Free Strategy Acquisition

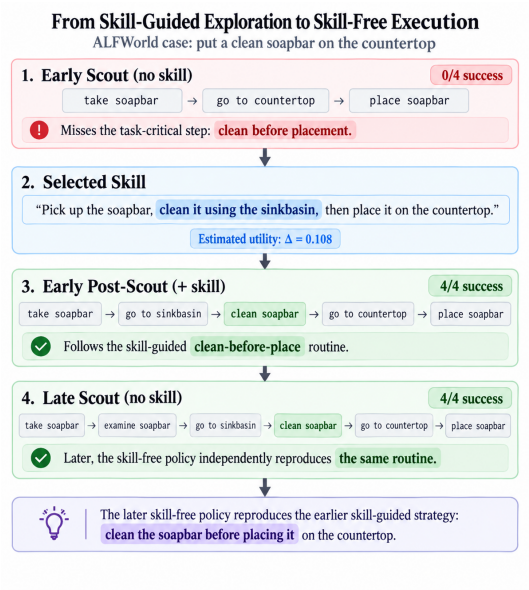



**Figure 13. From skill-guided exploration to skill-free execution.** An ALFWorld example in which a selected skill introduces the missing clean-before-place strategy, which is later reproduced without skill input.

Figure 13 shows that the early skill-free Scouts omit the required cleaning step, whereas the selected positive-utility skill enables all Post-Scouts to complete the task. At a later checkpoint, the skill-free policy independently reproduces the same clean-before-place routine, illustrating that the learned policy later executes the strategy without external skill input.

## D.2 ALFWorld: Clean Before Placement

**Task:** "put a clean soapbar in countertop." The selected skill inserts cleaning between retrieval and placement.

**1. Skill generation prompt (abridged template)**

*Repeated constraints and the full teaching example are omitted for space.*

**System message**

You extract reusable decision rules for an ALFWorld policy from four independent attempts at the same task. Write one concise skill that guides a new attempt using its current observations. Improve executable decisions rather than reproducing one completed trajectory.

**Evidence and comparison**

- The task defines the required object, quantity, state, and destination. Treat observations, inventory, admissible actions, and environment feedback as evidence; treat the agent's reasoning as an unverified belief.
- SUCCESS/FAILURE labels describe the final outcome only. Compare local decisions against their visible preconditions and feedback, preserving valid steps even when the episode fails.
- Ground corrections in rejected actions, confirmed state changes, or missing observations. For all-failure groups, retain supported progress and recovery rules without presenting an unobserved complete solution as verified.

**Abstraction and executability**

**1. Skill generation prompt (abridged template) (continued)**

- Express the target object, source, destination, required state, and appliance as semantic roles, then bind them to observed identifiers during execution. Track the same object instance across retrieval, transformation, and placement.
- Do not copy numbered objects, receptacles, fixed routes, or room layouts. Use only action forms supported by the interface, and verify explicit state changes such as cleaning, cooling, or heating before placement.
- Distinguish visiting a location from inspecting its contents. A failed search does not prove absence, and a rejected action should not be repeated without checking its command or preconditions.

**Output discipline**

Return exactly one concise card enclosed by `<skill>` and `</skill>`. Each rule should connect an observable condition to an action, verification check, or recovery step. Do not mention trajectories, outcome labels, rewards, demonstrations, or this prompt in the card.

**User message**

**Abridged teaching example—all attempts fail**

The attempts show three recurring local errors: treating a closed container as searched, using an identifier inconsistent with the observation, and repeating placement while still at the source. The example skill therefore recommends inspecting hidden contents, binding commands to observed identifiers, reading rejection feedback, and correcting the relevant precondition before retrying.

**Real task to process**

Task: {task}

Four independent skill-free attempts with environment outcomes:

<attempt_records>

{trajectories}

</attempt_records>

Write one coherent skill grounded in these records. Keep it within 400 English words and use exactly the following headings:

**# Skill: <short-kebab-case-name>**

**## When to apply**

**## Success-preserving strategy**

**## Failure patterns to suppress**

**## Procedure**

**## Decision rules**

**## Verification and recovery**

**## Stop conditions**

Return only `<skill>...</skill>` with the headings above.

**Assistant generation prefix**

<think>

</think>

**2. Selected skill (complete card)**

**# Skill: locate-and-clean-soapbar-before-placement**

**## When to apply** Use when the task requires locating a soapbar, cleaning it, and placing it on a countertop. The soapbar must be identified, retrieved, and cleaned before placement.

**## Success-preserving strategy** Preserve confirmed local progress: keep track of locations inspected, objects found, and actions taken. Distinguish between observed states and unverified assumptions.

**## Failure patterns to suppress** Do not assume the soapbar is absent after searching common locations. Do not attempt placement without confirming the soapbar is clean. Avoid repeating the same action without correcting its preconditions.

**## Procedure** 1. Inspect common locations for the soapbar, including sinkbasin, cabinets, drawers, and shelves.

2. Once found, pick up the soapbar and clean it using the sinkbasin.

3. After cleaning, move to the designated countertop and place the soapbar there.

**## Decision rules** - If the soapbar is not found in common locations, expand the search to less obvious places like the garbagecan.

**2. Selected skill (complete card) (continued)**

- If the soapbar is found but not clean, use the sinkbasin to clean it before attempting placement.
- If placement is rejected, verify that the soapbar is clean and located at the correct destination.

**## Verification and recovery** Check the feedback after each action to confirm state changes. If the soapbar is not clean, retry the cleaning step. If placement fails, recheck the soapbar's state and location.

**## Stop conditions** When the soapbar is clean and placed on the countertop, the task is complete. Budget expiry or incomplete search does not prove absence; continue searching until the soapbar is found and placed.

**3. Skill-guided trajectory (one Post-Scout; key turns)**

*[Turns 0–19: search omitted. Original turn indices are zero-based.]*

**Turn 20.** `take soapbar 1 from toilet 1`

*Observation:* You pick up the soapbar 1 from the toilet 1.

**Turn 21.** `go to sinkbasin 1`

*Observation:* You arrive at sinkbasin 1. On the sinkbasin 1, you see nothing.

**Turn 22.** `clean soapbar 1 with sinkbasin 1`

*Observation:* You clean the soapbar 1 using the sinkbasin 1.

**Turn 23.** `go to countertop 1`

*Observation:* You arrive at countertop 1. On the countertop 1, you see a toiletpaper 1.

**Turn 24.** `move soapbar 1 to countertop 1`

*[Terminal metadata: won=true, done=true. No subsequent textual observation was saved.]*

### D.3 WebShop: Verify the Required Option Before Purchase

**Task:** "i need a button down shirt with a long sleeve for a evereday wear medium size with v neck, and price lower than 50.00 dollars." This example uses the anchor-based variant and its actual successful Scout as Generator evidence.

**1. Skill generation prompt (complete template)**

**System message**

You are the Skill Extractor in a reinforcement-learning WebShop agent. Given one trajectory that the environment confirms succeeded, extract exactly one reusable policy skill grounded only in the supplied evidence. Success does not imply that every action in the trajectory was useful or optimal.

**User message**

You are the Skill Extractor in a reinforcement-learning WebShop agent. Extract the single most reusable decision mechanism supported by the private successful evidence below.
Private task: {task_description} Private trajectory: {trajectory}
STRICT SERIALIZATION CONTRACT — violation makes the entire output invalid:
1. The first character of your response must be '{' and the last character must be '}'.
2. Emit exactly one valid JSON object with exactly one key named 'skill'.
3. Never emit backticks, a code fence, analysis, preface, or trailing text.
4. Encode newlines inside the JSON string normally; ensure 'json.loads' can parse the complete response.
The 'skill' value is a 50-150 word Markdown card with headings in this exact order: '# Skill:', '## When to Use', '## Inputs to Identify', '## Workflow', '## Decision Rules'. Use no more than six complete sentences; list fragments are allowed.
Choose exactly one mechanism name: 'query-reformulation-after-weak-retrieval', 'explicit-contradiction-screening', 'page-evidence-constraint-verification', 'required-option-verification-before-purchase', 'compatibility-evidence-verification', or 'budget-and-constraint-gating'.
Use only these semantic role labels for request-specific concepts: product type, hard constraint, contradictory evidence, required option, compatibility requirement, page evidence, budget, candidate, search result. Keep them verbatim and never add a concrete example or subtype. Invalid: 'device compatibility', 'device family', 'color or size', 'fragrance', 'power rating', 'running use'. Valid replacements: 'compatibility requirement', 'required option', or 'hard constraint'. Do not output any product/category name, brand, identifier, query phrase, number, currency amount, size, color, material, audience, device, power rating, scent, skin property, sport/use case, or requested attribute. Do not mention private evidence, task, source, trajectory, example, outcome, reward, or success.
Describe only the selected mechanism: its observable trigger, ordered action, and verification or stopping condition. Do not summarize the full episode, invent controls or facts, or claim universal optimality. Silently verify that the card would remain valid for a different product category without editing.
Your response must begin exactly like this (with the selected name replacing the placeholder): {"skill":"# Skill: mechanism-name\n

**Assistant generation prefix**

<think>
</think>

**2. Selected skill (complete card)**

**# Skill: required-option-verification-before-purchase**
**## When to Use** When a product meets all hard constraints but lacks a required option.
**## Inputs to Identify** A product that satisfies price, type, and compatibility requirements but is missing a required option.
**## Workflow** Check if the required option is available in the specified size or color.
**## Decision Rules** If the required option is available, proceed with purchase; otherwise, backtrack to refine search.

**3. Skill-guided trajectory (one Post-Scout; all four actions)**

*[Reasoning omitted; page observations excerpted. Turn indices are zero-based.]*

**Turn 0.** `search[button down shirt long sleeve v neck medium]`

*Observation:* . . . B09KR6LR84 [SEP] Shirt for Women Button Down Tops Colorful Blouses Casual V Neck Sexy Long Sleeve T-Shirts [SEP] $25.99 . . .

**Turn 1.** `click[b09kr6lr84]`

*Observation:* . . . size [SEP] small [SEP] medium [SEP] large [SEP] xx-large [SEP] 3x-large . . . Price: $25.99 . . .

**Turn 2.** `click[medium]`

*Observation:* . . . size [SEP] small [SEP] medium [SEP] large [SEP] xx-large [SEP] 3x-large . . . Price: $25.99 . . .

**Turn 3.** `click[buy now]`

*[Terminal metadata: won=true, done=true, task_score=1. No subsequent textual observation was saved.]*

### D.4 ScienceWorld: Complete the Circuit Before Classifying

**Task (excerpt):** "Your task is to determine if metal fork is electrically conductive." The task assigns conductive objects to the red box and nonconductive objects to the green box.

**1. Skill generation prompt (abridged template)**

*Repeated constraints and the full teaching example are omitted for space.*

**System message**

You extract reusable ScienceWorld laboratory skills from four attempts at the same task. Learn observable decision rules from successful and unsuccessful attempts, and use only the supplied evidence. A failed or budget-limited attempt may still contain valid partial progress.

Never invent instruments, commands, object properties, causal effects, or experimental results. In particular, an absent signal is not a negative material test until the observed setup is complete and verified. Missing equipment, invalid commands, incomplete circuits, and action limits must not be converted into scientific conclusions.

Abstract exact object instances, rooms, colors, and destinations into roles read from the current task. Preserve scientific properties and action forms only when they are supported by the observed attempts.

Return one evidence-grounded card of at most 400 English words, enclosed by `<skill>` and `</skill>`.

**User message**

Task: `{task}`

Four observed attempts with environment-confirmed outcomes:

`{trajectories}`

Produce a reusable procedure rather than a transcript tied to one variation. Preserve verified progress, identify supported failure-producing decisions, and bind commands to currently observable objects. Distinguish measured facts from hypotheses, and check equipment, material state, and prerequisites before interpreting an unsuccessful experiment.

**Abridged teaching example—verify before classifying**

For a conductivity task with no clear signal, first inspect the available equipment and connections. Correct only problems supported by feedback, then retest and classify the material only when the task's observable criterion is met. If the setup remains incomplete, retain uncertainty rather than assigning an unsupported material property.

Use exactly the following structure:

`<skill>`

**# Skill: short reusable mechanism name**
**## When to apply**
**## Success-preserving strategy**
**## Failure patterns to suppress**
**## Procedure**
**## Decision rules**
**## Verification and recovery**
**## Stop conditions**

`</skill>`

Output only the card. Do not mention rewards, trajectories, demonstrations, or fixed private box colors and room names.

**Assistant generation prefix**

`<think>`
`</think>`

**2. Selected skill (complete card)**

**# Skill: Test Conductivity Using Available Setup**

**## When to apply** A material is suspected to be electrically conductive, and a test setup involving a power source and a light bulb is available.

**## Success-preserving strategy** Maintain the material in the inventory, confirm the presence of a power source and light bulb, and ensure the setup is connected correctly. Use the task's specified destination for classification.

**## Failure patterns to suppress** Using unverified equipment, attempting to use non-existent tools like a conductivity tester, or connecting the material without a complete circuit.

**2. Selected skill (complete card) (continued)**

**## Procedure** 1. Focus on the metal fork to examine its properties.
2. Pick up the metal fork to place it in the inventory.
3. Check for the presence of a power source and light bulb in the environment.
4. Connect the metal fork to the power source and light bulb to form a circuit.
5. Observe if the light bulb illuminates to determine conductivity.
6. If the light bulb lights up, place the metal fork in the destination specified by the current task.

**## Decision rules** If the light bulb does not illuminate, check the connections and ensure the circuit is complete. If the power source or light bulb is not available, do not proceed with the test. If the connection command is rejected, verify the presence of the required objects before retrying.

**## Verification and recovery** Confirm that the light bulb lights up as a clear signal of conductivity. If the light bulb does not respond, check the setup for errors without assuming a material property. If the setup is confirmed but the signal is absent, retain uncertainty and avoid classification.

**## Stop conditions** Stop when the light bulb illuminates, confirming conductivity, or when the setup is confirmed to be incomplete and no further actions can be taken.

**3. Skill-guided trajectory (one Post-Scout; key turns)**

*[Turns 0–10: search, pickup, and navigation omitted. Zero-based indices.]*

**Turn 11.** `connect metal fork to battery`

*Observation:* Connections must specify specific points of contact on polarized electrical components (e.g. anode, cathode)

**Turn 12.** `connect metal fork to anode in battery`

*Observation:* terminal 1 on metal fork is now connected to anode on battery

**Turn 13.** `connect metal fork to blue light bulb cathode`

*Observation:* terminal 2 on metal fork is now connected to cathode on blue light bulb

**Turn 14.** `look at blue light bulb`

*Observation:* a blue light bulb, which is off. its anode is connected to: nothing. its cathode is connected to: a terminal 2 on metal fork.

*[Turns 15–16: inspections omitted.]*

**Turn 17.** `connect battery cathode to blue light bulb anode`

*Observation:* cathode on battery is now connected to anode on blue light bulb

**Turn 18.** `look at blue light bulb`

*Observation:* a blue light bulb, which is on. its anode is connected to: a cathode on battery. its cathode is connected to: a terminal 2 on metal fork.

**Turn 19.** `move metal fork to red box`

*[Terminal metadata: won=true, done=true, task_score=100. No subsequent textual observation was saved.]*

### D.5 $\tau^2$-Bench: Confirm an Exchange Before Executing It

**Task (summary):** Exchange a delivered action camera for an available 4K, waterproof, black variant of the same product.

**1. Skill generation prompt (complete template)**

**System message**

You are a strategy extractor for a tool-using customer-service agent. You receive the exact agent-visible context and four independent attempts for one task. Infer one reusable strategy that preserves mechanisms shared by good attempts and suppresses mechanisms associated with bad attempts. The domain policy and tool schemas always override the strategy. Never reveal or copy task-specific entities. Return only the requested Markdown skill card and nothing else.

**User message**

DOMAIN: retail

**RETAIL FOCUS**

- Distinguish order/item lookup from return, exchange, refund, address, payment, and cancellation writes.
- Verify identity, order status, item eligibility, delivery state, quantities, refund destination, replacement availability, and confirmation boundaries before irreversible actions.
- Handle partial orders and multi-item requests explicitly; stop or escalate when policy forbids an action.
- Keep order IDs, product IDs, names, addresses, dates, amounts, and payment IDs abstract.
- Do not name a product category or repeat a user's subjective product preference; say 'item' and 'requested variant' so the mechanism transfers across products.

```
<AGENT_VISIBLE_CONTEXT>
{visible_context}
</AGENT_VISIBLE_CONTEXT>
<ATTEMPT index=0 outcome={outcomes[0]}>
{attempts[0]}
</ATTEMPT>
<ATTEMPT index=1 outcome={outcomes[1]}>
{attempts[1]}
</ATTEMPT>
<ATTEMPT index=2 outcome={outcomes[2]}>
{attempts[2]}
</ATTEMPT>
<ATTEMPT index=3 outcome={outcomes[3]}>
{attempts[3]}
</ATTEMPT>
```

**OUTPUT CONTRACT**

Return the skill card directly as Markdown, beginning with '# Skill:' and containing these sections in order: '## When to apply', '## Policy and tool checks', '## Procedure', '## Decision rules', '## Verification and recovery', and '## Stop or escalation conditions'. Do not wrap it in JSON or a code fence.

The skill should be 120-240 English words, with no more than two concise sentences or bullets per section. It must be operational rather than generic and reusable on a different query in the same domain. Stable tool names and argument roles are allowed. Do not copy any customer name, account/order/reservation/payment identifier, email, phone number, address, route, flight number, date, price, exact task value, or hidden evaluator language. Do not compare or refer to the supplied examples, their outcome labels, trajectories, rewards, Delta, demonstrations, or grading. Do not solve the concrete query. Extract one mechanism, not a broad customer-service handbook.

**FINAL RETAIL CHECK**

Recheck that identifiers and product preferences are abstract, every mutation has its required status/eligibility/confirmation checks, and the card transfers to another item or order.

**2. Selected skill (complete card)**

**# Skill: Exchange item for same product type in delivered order**

**## When to apply** Use to exchange an item from a delivered order for a different variant of the same product type.

**2. Selected skill (complete card) (continued)**

**## Policy and tool checks** - Verify order is delivered
- Confirm exchangeable variant is available
- Request refund method (original payment or gift card)
- Ensure only one exchange per order

**## Procedure** 1. Fetch user and order details
2. Identify target item and eligible variants
3. Present options and price difference
4. Confirm exchange details before proceeding

**## Decision rules** - Reject exchanges to different product types
- Prevent multiple exchanges per order
- Escalate if payment method incompatibility detected

**## Verification and recovery** - Confirm refund method matches eligibility
- Store exchange details in order history
- Provide return instructions via email

**## Stop or escalation conditions** Transfer if request exceeds exchange scope (e.g., full order modifications).

**3. Skill-guided trajectory (one Post-Scout; key turns)**

*[Identity lookup and other dialogue omitted. Assistant turns are 1-based; tool results are field excerpts.]*
**Turn 4.** `{"name": "get_order_details", "arguments": {"order_id": "#W6759506"}}`
*Observation:* `... "status": "delivered" ...`
**Turn 5.** `{"name": "get_product_details", "arguments": {"product_id": "3377618313"}}`
*Observation:* `... "6700049080": {"item_id": "6700049080", "options": {"resolution": "4K", "waterproof": "yes", "color": "black"}, "available": true, "price": 466.75} ...`
**Turn 7. Agent (excerpt):**
**Price difference:** -$36.76 (you'll receive a refund)
**Refund method:** Gift Card (current balance: $200.00)
. . . Do you confirm this exchange? Please reply **yes** or **no**.
*User:* Yes, I confirm the exchange.
**Turn 8.** `{"name": "exchange_delivered_order_items", "arguments": {"new_item_ids": ["6700049080"], "payment_method_id": "gift_card_4444444", "item_ids": ["5925362855"], "order_id": "#W6759506"}}`
*Observation:* `... "status": "exchange requested" ... "exchange_price_difference": -36.76 ...`
*[Recorded outcome: reward=1, database match=true. This confirms an exchange request, not physical return or refund completion.]*