TRACKABLE SKILL LIBRARY LIFECYCLE MANAGEMENT FOR AGENTIC REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Skill-augmented agentic reinforcement learning (RL) has emerged as an effective approach to improving LLM agents on long-horizon tasks. Despite this progress, individual skills may follow distinct trajectories as the policy evolves, with some becoming redundant after the model internalizes their capabilities, others remaining valuable as external scaffolding, and still others turning harmful. However, existing skill library systems fail to identify each skill's true contribution due to free-riding among co-retrieved skills, difficulty confounding across tasks, and the absence of skill-specific counterfactual evidence, which prevents reliable skill lifecycle management. We propose TrackRL, an agentic RL framework in which skill tracking drives full lifecycle management of the skill library. Specifically, within each GRPO group, we turn grouped sampling into an online randomized controlled experiment. Independent random injection and within-group contrasts on the same task isolate each skill's true contribution from co-retrieved skills and task difficulty, and reliability-weighted aggregation accumulates this evidence across groups. As evidence accumulates, we continuously track skill utility, which measures the skill's causal contribution as external scaffolding, and model mastery, which reflects how far the corresponding capability has been internalized. Along this line, we formulate skill lifecycle management as an evidence-gated state-transition process that governs each skill from generation and admission through protected incubation to retention, retirement, or removal, while a soft capacity bound controls library growth. Extensive experiments show that TrackRL outperforms recent skill-augmented RL baselines in both library-assisted and skill-free evaluation, with further analyses revealing how skill value shifts during policy learning and how evidence-driven state transitions shape skill library evolution. Our code is now available at https://anonymous.4open.science/r/TrackRL-8D6D.

1 INTRODUCTION

Agentic reinforcement learning (RL) has substantially improved LLM agents on long-horizon tasks, including household planning (Shridhar et al., 2021), web navigation (Yao et al., 2022), and interactive coding tasks (Trivedi et al., 2024). Existing skill-augmented approaches optimize model policies through RL while distilling experience into reusable skills stored in an external library (Xia et al., 2026a; Yao et al., 2026). As the library grows, relevant skills are retrieved as context to guide subsequent policy updates, enabling policy–library co-evolution (Shi et al., 2026; Li et al., 2026b).

This co-evolution dynamically couples the model's internal policy with its external skill library. A skill derived from reflection on a failed trajectory fills a capability gap at one stage of training, but that gap may narrow or shift as the policy evolves. As RL continually reshapes model capabilities and, in turn, revalues existing skills (Yu et al., 2026b;a), an effective skill library should evolve with the policy it supports, rather than be constructed and maintained in isolation (He et al., 2026).

Individual skills follow distinct evolutionary trajectories. As the policy evolves, some skills lose marginal value and become redundant once the model internalizes the capability (Xiao et al., 2026; Xia et al., 2026b). Others remain valuable as external scaffolding for long-tail cases or hard-to-internalize knowledge (Wang et al., 2024; Suzgun et al., 2025), while some become harmful due to reflection errors or policy changes (Liu, 2026; Zhang & Li, 2026). To manage these divergent trajec-

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.

tories, a continually learning agent must track each skill's current utility and the model's mastery of the corresponding capability, and use the accumulated evidence to govern each skill's full lifecycle, keeping the library aligned with the evolving policy (Pu et al., 2026; Lin et al., 2026b).

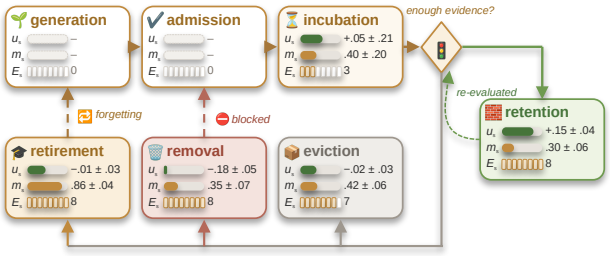


Figure 1: **Skill lifecycle.** Tracked utility, mastery, and evidence drive the lifecycle of each skill. Retained skills serve as external scaffolding, whereas retired skills have been internalized by the policy.

However, today's skill library systems cannot identify what an individual skill contributes or whether the current model still needs it. First, a successful rollout typically relies on multiple retrieved skills, yet its episode-level outcome is attributed indiscriminately to all of them (Tu et al., 2026; Li et al., 2026a), allowing non-contributing skills to free-ride on those that actually drive success. This ambiguity can be amplified when the library contains both task-level planning skills and step-level execution skills, as each action draws on a different set of step-level skills, but all such sets are still evaluated against a single episode-level outcome (Wang et al., 2026a). Second, because skills are retrieved rather than randomly assigned (Xia et al., 2026a; Wang et al., 2023), a skill's observed utility depends partly on the difficulty of the tasks it serves, a phenomenon we term difficulty confounding. Successes on easy tasks may inflate its apparent contribution, while failures on hard tasks may obscure meaningful assistance. Most critically, existing pipelines lack the skill-specific counterfactual evidence needed to estimate a skill's marginal utility and the model's mastery of the corresponding capability (Hu et al., 2026), leaving its evolving value difficult to assess. We therefore need a debiased, longitudinal estimator that tracks each skill's utility across task types and co-retrieved skill sets, together with the model's mastery of the corresponding capability, to provide an evidence-based foundation for skill library lifecycle management.

Even as skill-level evidence accumulates, existing frameworks lack a principled mechanism for translating it into reliable skill lifecycle decisions. On the one hand, they address capacity constraints through utility-based pruning, without protecting skills whose limited evidence may conceal latent value during incubation. On the other hand, they fail to distinguish the diversity of skill lifecycle states and the complexity of transitions between them. Each skill follows a distinct life course: born from reflection, incubated as evidence accumulates, evolves through repeated retrieval and use, and may eventually retire once the model internalizes the corresponding capability (Deng et al., 2024), persist as an external scaffold (Suzgun et al., 2025), or be removed if it becomes harmful. We therefore need trackable, state-driven skill lifecycle management that uses skill utility and model mastery to identify each skill's evolving state, gates every transition on sufficient evidence, and guides its progression from generation and admission through protected incubation to retention, retirement, or removal.

We propose **TrackRL**, an agentic RL framework in which skill tracking drives full lifecycle management of the skill library, closing the loop of policy–library co-evolution. Building on the standard skill-augmented RL pipeline, TrackRL turns the grouped sampling of agentic RL into an online randomized controlled experiment by independently randomizing the injection of each selected skill within a group. The resulting within-group contrasts yield two quantities that jointly characterize the evolving state of a skill, namely skill utility, its causal contribution as external scaffolding, and model mastery, the extent to which its capability has been internalized, while addressing free-riding and difficulty confounding by design. Grounded in this evidence, we formulate skill library lifecycle management as an evidence-gated state-transition process. Using tracked utility, mastery, and accumulated evidence, we govern each skill from generation and admission through protected incubation to retention, retirement, or removal under a soft capacity, with each transition supported by sufficient evidence. Experiments on ALFWorld and AppWorld show that TrackRL outperforms recent skill-augmented RL baselines in both skill-assisted and skill-free evaluation, and further analyses reveal how skill value shifts as the policy learns and how evidence-driven transitions shape the evolution of the library. Extensive experiments show that TrackRL outperforms recent skill-augmented RL baselines in both library-assisted and skill-free evaluation. Further analyses demonstrate how skill value shifts as the policy learns and how evidence-driven state transitions shape library evolution.

2 TRACKRL

2.1 SKILL-AUGMENTED RL WITH SKILL TRACKING

Skill Organization. LLM agents require both task-level planning and step-level execution to solve long-horizon tasks. Following prior work (Tu et al., 2026; Agashe et al., 2024; Wang et al., 2025), we adopt a hierarchical skill organization and maintain two initially empty textual skill pools, a task-level skill pool $\mathcal{P}_{\text{task}}$ and a step-level skill pool $\mathcal{P}_{\text{step}}$. Specifically, task-level skills generalize across tasks with similar planning structures and guide episode-level decisions, whereas step-level skills generalize across similar local situations and guide action selection from the current observation.

Skill Retrieval and Reranking. From the skill library, the agent must select a few skills that are both relevant to the current query and useful under the current model policy. Accordingly, we retrieve candidates by semantic similarity and rerank them with accumulated usage evidence. Each skill s is associated with a retrieval key $\kappa(s)$, a short description of the context in which it applies. We retrieve task-level skills once at the beginning of each episode using the task description as query x, and step-level skills at every turn the task description concatenated with the current observation. Formally, we encode the query and retrieval keys through a text embedding model, and retrieve at most m candidates from the corresponding pool $\mathcal{P}$,

$$\mathcal{R}(x;\mathcal{P}) \;=\; \mathrm{Top}_m\big\{\, s\in\mathcal{P} \;\big|\; \mathrm{sim}\big(x,\kappa(s)\big)\geq\delta \,\big\}, \tag{1}$$

where $\mathrm{sim}(x,\kappa(s))$ is the cosine similarity and δ is a threshold. Retrieval identifies semantically relevant skills, but relevance alone does not show whether a skill helps or harms the current policy. Thus, we rerank the candidates by combining semantic relevance and usage evidence,

$$\mathrm{score}(s) \;=\; \alpha\,\widetilde{\mathrm{sim}}\big(x,\kappa(s)\big) \;+\; (1-\alpha)\left[u_s+\eta\sqrt{\frac{\log\big(1+N_{\mathcal{P}}\big)}{1+N_s}}\right], \tag{2}$$

where $\widetilde{\mathrm{sim}}=(\mathrm{sim}+1)/2$ maps cosine similarity to $0,1$, u_s is the current utility estimate, N_s counts episodes in which s was selected, and $N_{\mathcal{P}}$ is the total selection count across the pool. The coefficient α balances semantic relevance against usage evidence, while η controls an exploration bonus that decreases with exposure, giving less frequently selected skills opportunities to be evaluated before their utility estimates become reliable (Auer et al., 2002). Finally, we select the K highest-scoring candidates for prompt injection. We keep task-level skills active throughout the episode, whereas we append step-level skills after the current observation to guide only the next action generation.

Skill-Augmented Policy Optimization. We optimize the policy using standard GRPO (Shao et al., 2024). The policy π_θ solves each task through multi-turn interaction, with assigned skills provided as context at each turn. Each episode e produces a binary outcome $y_e\in 0,1$ and receives reward $R_e=y_e-\nu\phi_e$, where ϕ_e is the fraction of invalid actions. Unlike methods that give additional rewards for skill-conditioned episodes (Xia et al., 2026a; Tu et al., 2026), we omit such rewards, as they may encourage reliance on skills rather than task completion. At each training step, we sample M tasks, generate a group g of B episodes per task using the behavior policy π_{old}, and maximize

$$\mathcal{J}(\theta)=\mathbb{E}_g\Big[\tfrac{1}{B}\textstyle\sum_{e\in g}\frac{1}{L_e}\sum_{k=1}^{L_e}\min\big(\rho_{e,k}A_e,\;\mathrm{clip}(\rho_{e,k},1{-}\epsilon,1{+}\epsilon)\,A_e\big)\Big]-\beta\,D_{\mathrm{KL}}\big(\pi_\theta\,\|\,\pi_{\text{ref}}\big), \tag{3}$$

with

$$\rho_{e,k}=\frac{\pi_\theta(o_{e,k}\mid h_{e,k})}{\pi_{\text{old}}(o_{e,k}\mid h_{e,k})},\qquad A_e=\frac{R_e-\mathrm{mean}_{e'\in g}\,R_{e'}}{\mathrm{std}_{e'\in g}\,R_{e'}},$$

where L_e is the number of generated tokens in episode e, $h_{e,k}$ is the context preceding token $o_{e,k}$, and ϵ and β set the clipping range and the KL weight. In standard GRPO, rollouts within a group share the same task and policy, forming repeated trials under otherwise matched conditions. In skill-augmented RL, this structure motivates us to vary skill assignments across these rollouts so that their episode-level outcomes can reveal each skill's contribution.

Skill Randomized Injection. To identify individual skill contributions from episode-level outcomes, we independently randomize the injection of each selected skill across episodes within a GRPO group. Accordingly, for each skill s selected for episode e, we sample

$$z_{s,e}\sim\mathrm{Bernoulli}(p),\qquad z_{s,e}\perp z_{s',e}\quad\forall s\neq s'. \tag{4}$$

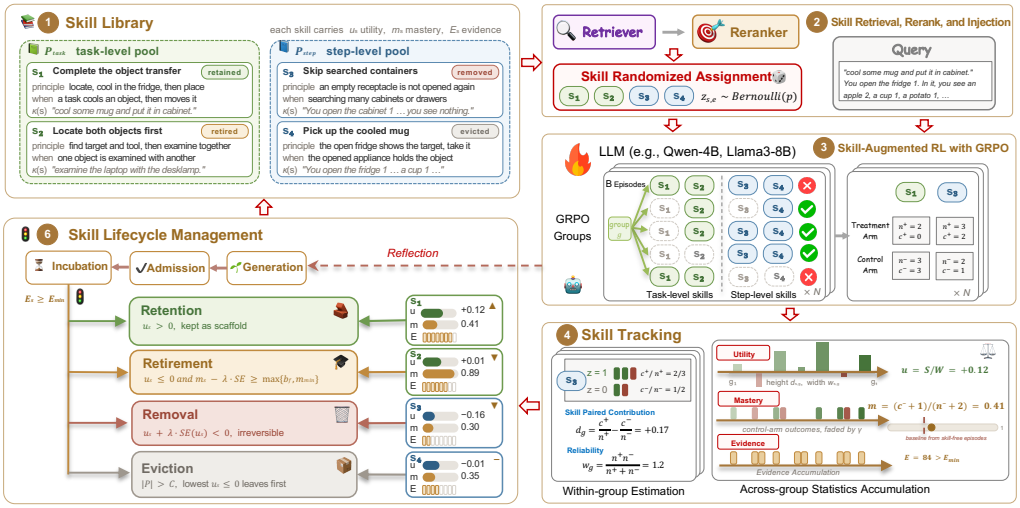


Figure 2: **Overview of TrackRL.** We maintain a library of task-level and step-level skills, which we retrieve and rerank before randomizing their injection within GRPO groups to create per-skill treatment and control arms. From these comparisons, we track skill utility, model mastery, and evidence to guide skill lifecycle management. The updated library then supports subsequent policy optimization, closing the loop of policy–library co-evolution.

For each skill s selected after retrieval and reranking, episodes with $z_{s,e} = 1$ receive the skill and form the treatment arm, whereas those with $z_{s,e} = 0$ withhold it and form the control arm. Consequently, each group becomes a set of concurrent randomized controlled experiments over multiple selected skills without additional rollouts. Randomized injection thus compares the two arms under the same task and policy, removing difficulty confounding, while independent assignments across skills prevent free-riding. In addition, each group reserves one task-level and one step-level skill-free episode, each omitting all corresponding skills to measure the policy's unaided performance. To assign each episode unambiguously to the treatment or control arm for every selected skill, we sample each task-level assignment once at episode onset and each step-level assignment upon first retrieval, reusing the latter whenever the same skill is retrieved again.

Skill Utility for the Current Policy. To quantify each skill's contribution to the current policy, we define its utility u_s as the change in success rate caused by injecting rather than withholding it. We estimate the skill utility u_s from within-group contrasts accumulated over training. For skill s in group g, let $n^+_{s,g}$ and $n^-_{s,g}$ denote the numbers of episodes in the treatment and control arms, and $c^+_{s,g}$ and $c^-_{s,g}$ their respective numbers of successes. Whenever both arms are nonempty, the group contributes a comparison $d_{s,g}$ with weight $w_{s,g}$,

$$d_{s,g} = \frac{c^+_{s,g}}{n^+_{s,g}} - \frac{c^-_{s,g}}{n^-_{s,g}}, \qquad w_{s,g} = \frac{n^+_{s,g}\, n^-_{s,g}}{n^+_{s,g} + n^-_{s,g}}. \tag{5}$$

Because the two arms share the same task and policy, $d_{s,g}$ isolates the effect of skill injection from task difficulty. The weight $w_{s,g}$ reflects the reliability of this contrast, giving more weight to comparisons with more episodes and a more balanced split between the arms. As the policy evolves, earlier comparisons become less representative of a skill's current utility, so we discount past evidence by $\gamma \in (0, 1]$ and update four running statistics at each training step,

$$\begin{aligned} W_s &\leftarrow \gamma W_s + \textstyle\sum_{g \in \mathcal{G}_s} w_{s,g}, & S_s &\leftarrow \gamma S_s + \textstyle\sum_{g \in \mathcal{G}_s} w_{s,g}\, d_{s,g}, \\ Q_s &\leftarrow \gamma Q_s + \textstyle\sum_{g \in \mathcal{G}_s} w_{s,g}\, d^2_{s,g}, & G_s &\leftarrow \gamma G_s + |\mathcal{G}_s|, \end{aligned} \tag{6}$$

where $\mathcal{G}_s$ denotes the groups at the current training step in which both arms are non-empty. W_s accumulates the weights, S_s the weighted comparisons, Q_s their weighted squares, and G_s the effective number of comparisons. These statistics yield the utility estimate and its standard error,

$$u_s = \frac{S_s}{W_s}, \qquad \mathrm{SE}(u_s) = \sqrt{\frac{Q_s - S_s^2/W_s}{W_s\,(G_s - 1)}}, \qquad G_s \geq 2. \tag{7}$$

The estimate u_s captures the skill's current marginal contribution, while $\mathrm{SE}(u_s)$ reflects variation across its comparisons. Since each group contains only B episodes, individual comparisons can be

coarse and noisy, making uncertainty essential to avoid relying on a point estimate alone for skill lifecycle decisions. We additionally maintain an undiscounted evidence count $E_s \leftarrow E_s + |\mathcal{G}_s|$.

Model Mastery of Skill Capabilities. Conversely, to assess whether the model has internalized the capability carried by skill s, we estimate mastery m_s from the current policy's success rate when s is selected but withheld. Utility alone cannot reveal why a skill contributes little. For example, near-zero utility may indicate either an ineffective skill or a policy that already succeeds without it, with opposite implications for the skill's lifecycle. Control-arm success distinguishes these cases, so we accumulate the corresponding episode and success counts across groups using the same discount,

$$n_s^- \;\leftarrow\; \gamma\, n_s^- + \textstyle\sum_g n_{s,g}^-, \qquad c_s^- \;\leftarrow\; \gamma\, c_s^- + \textstyle\sum_g c_{s,g}^-. \tag{8}$$

Starting with a uniform prior, we use these counts to obtain a posterior $\mathrm{Beta}(c_s^- + 1,\; n_s^- - c_s^- + 1)$, whose mean and standard deviation give the mastery estimate and its uncertainty,

$$m_s \;=\; \frac{c_s^- + 1}{n_s^- + 2}, \qquad \mathrm{SE}(m_s) \;=\; \sqrt{\frac{m_s\,(1 - m_s)}{n_s^- + 3}}, \tag{9}$$

where m_s starts at a neutral 0.5 and approaches the observed success rate as evidence accumulates. To keep mastery a reading of the model's capability, we define it only for task-level skills. While success without a task-level skill suggests that the policy has internalized the capability required for the task, success without a step-level skill may instead occur because the skill was nonessential at that step or that later actions or other skills compensated for its absence.

2.2 Skill Library Full Lifecycle Management

Skill Generation, Admission, and Protection. We generate each skill from failure, admit non-redundant candidates, and protect new skills until they accumulate enough evidence for lifecycle decisions. The reflection model extracts task-level and step-level skills from failed trajectories in the groups with the lowest success rates. Since recurring failure patterns can produce near-duplicate skills that dilute the tracking evidence and undermine reliable utility and mastery estimates, deduplication check not only control library size but also preserve reliable statistics. A candidate s_{new} is admitted only if it is dissimilar to all active skills and all previously removed harmful skills,

$$\max_{s \in \mathcal{P} \cup \mathcal{H}} \mathrm{sim}(s_{\mathrm{new}}, s) \;<\; \delta_{\mathrm{dup}}, \tag{10}$$

where $\mathcal{H}$ records skills removed as harmful. Once admitted, a skill starts with empty statistics and enters an incubation period, remaining protected from all exits while $E_s < E_{\min}$ or $\mathrm{SE}(u_s)$ is undefined, regardless of its point estimates. Here, $E_{\min}$ is the minimum evidence threshold.

Skill Removal, Retirement, and Retention. Once sufficient evidence accumulates, we remove harmful skills, retire internalized skills, and retain the rest. Specifically, a skill is deemed harmful and permanently removed once even the upper confidence bound on its utility falls below zero,

$$E_s \geq E_{\min} \;\wedge\; u_s + \lambda\, \mathrm{SE}(u_s) < 0, \tag{11}$$

whereas a task-level skill is retired only when its utility is non-positive and even the lower confidence bound on its mastery exceeds both the absolute threshold $m_{\min}$ and the family baseline $b_{f(s)}$,

$$E_s \geq E_{\min} \;\wedge\; u_s \leq 0 \;\wedge\; m_s - \lambda\, \mathrm{SE}(m_s) \geq \max\left\{b_{f(s)},\, m_{\min}\right\}, \tag{12}$$

where λ sets the confidence level of both one-sided tests and $b_{f(s)}$ denotes the discounted success rate of the task-level skill-free episodes of the same task type. Since task difficulty also inflates mastery, this baseline calibrates it to the task family, imposing a higher threshold on skills from easier families. If model forgetting makes a retired skill necessary again, subsequent failures may trigger its regeneration and readmission as a new candidate, whereas skills removed as harmful remain permanently excluded from the library. As the library converges, skills that retain positive utility without being internalized persist in the skill library as long-term scaffolding.

Soft Skill Library Capacity. The skill library usually requires an upper bound C on the size of each pool to control retrieval cost and storage. However, a hard bound evicts the lowest-ranked skills upon overflow regardless of their evidence, reintroducing the noise-driven pruning prevented by the protection rule. We therefore make C a soft bound. Whenever $|\mathcal{P}| > C$, eligible skills in $\{s : E_s \geq E_{\min} \wedge u_s \leq 0\}$ are evicted in ascending order of u_s, while the pool may remain temporarily above C if this set is exhausted. The capacity rule thus prioritizes the eviction of skills confidently deemed non-beneficial while protecting those still in incubation.

Table 1: **Results on ALFWorld.** SR (%) in the skill-assisted and skill-free modes on the same checkpoint, with per-class results on the seen split. *Origin* is the base model before RL, Luna and Flash denote the reflection model. "-" marks a method without a skill library. In each setting the best summary result is shown in **bold**.

Method	Seen, per task class						Seen skill-assisted	Seen skill-free	Unseen skill-assisted	Unseen skill-free
	Pick	Clean	Cool	Look	Heat	Pick2				
Closed-source LLMs										
Gemini-2.5-Flash	88.6	29.6	28.0	23.1	56.2	50.0	-	50.0	-	49.2
GPT-5.6-Luna	74.3	29.6	44.0	53.8	43.8	45.8	-	50.0	-	50.0
RL from the base model (Qwen3-4B-Instruct-2507)										
Origin	40.5	11.5	8.0	21.4	26.7	0.0	-	19.3	-	27.6
GRPO	82.9	55.6	52.0	53.8	56.2	37.5	-	58.6	-	53.0
SkillRL (Flash)	11.4	11.1	12.0	7.7	12.5	8.3	11.7	20.3	10.2	20.3
SkillRL (Luna)	82.9	92.6	60.0	46.2	75.0	0.0	62.1	42.1	58.2	45.5
D2Skill (Flash)	87.9	63.0	37.0	72.7	66.7	73.3	66.4	61.7	69.5	58.6
D2Skill (Luna)	80.7	83.3	18.5	41.2	63.6	50.0	57.0	52.3	54.7	52.3
TrackRL (Flash)	75.0	66.7	88.0	66.7	43.8	59.1	68.8	65.6	70.3	62.5
TrackRL (Luna)	89.7	79.3	50.0	76.9	100.0	50.0	**78.1**	**72.7**	**71.9**	**64.1**
RL from the SFT warm up (Qwen3-4B-Instruct-2507 + Luna SFT, Luna reflection)										
Origin + SFT	48.6	19.2	16.0	42.9	53.3	26.1	-	33.6	-	40.3
GRPO	100.0	100.0	92.0	78.6	100.0	82.6	-	93.6	-	91.8
SkillRL	97.1	96.3	84.0	92.3	87.5	87.5	91.4	85.0	82.8	71.6
D2Skill	100.0	100.0	90.0	100.0	93.3	94.4	96.9	93.8	96.1	91.4
TrackRL	100.0	100.0	96.2	88.9	100.0	100.0	**98.4**	**96.1**	**97.7**	**95.3**

3 EXPERIMENTS

We conduct experiments on ALFWorld (Shridhar et al., 2021) and AppWorld (Trivedi et al., 2024), reporting the Success Rate (SR) on ALFWorld and the Task Goal Completion (TGC) and Scenario Goal Completion (SGC) on AppWorld. We compare TrackRL against three groups of methods, namely (1) Proprietary LLMs, including Gemini-2.5-Flash and GPT-5.6-Luna under direct prompting, (2) Skill-free models, including the untrained base model and GRPO, and (3) Skill-augmented RL methods, including SkillRL (Xia et al., 2026a) and D2Skill (Tu et al., 2026). All trained methods use Qwen3-4B-Instruct-2507 as the policy model, and the skill-augmented methods use GPT-5.6-Luna and Gemini-2.5-Flash as reflection models. Full details of the benchmarks, evaluation protocol, baselines, and implementation are provided in Appendix A.

Main Results. To evaluate whether TrackRL trains stronger agents, we report the results on ALFWorld and AppWorld in Tables 1 and 2, where every trained method is evaluated in both the skill-assisted and skill-free modes on the same checkpoint. We make four observations. (1) TrackRL achieves the highest skill-assisted SR on ALFWorld. Under cold start with the Luna reflection model, TrackRL reaches 78.1% on the seen split and 71.9% on the unseen split, compared with 57.0% and 54.7% for D2Skill and 62.1% on the seen split for SkillRL, and under warm start it reaches 98.4% against 96.9% for D2Skill. (2) The advantage holds across reflection models. With Flash as the reflection model, TrackRL reaches 68.8% on the seen split and 70.3% on the unseen split, ahead of D2Skill (Flash) at 66.4% and 69.5% and far ahead of SkillRL (Flash) at 11.7% and 10.2%, so the two TrackRL variants occupy the first and second places in every skill-assisted column of this setting. (3) Skill-free evaluation measures what the policy has internalized. Under cold start TrackRL (Luna) retains 72.7% on the seen split and 64.1% on the unseen split without any skill injection, ahead of GRPO at 58.6% and 53.0% and of the strongest skill-augmented baseline D2Skill (Flash) at 61.7% and 58.6%; under warm start it retains 96.1% and 95.3% against 93.6% and 91.8% for GRPO. Tracking therefore yields a policy that is stronger even when the library is withheld, rather than one that merely leans on injected text. (4) The ordering carries over to AppWorld. Under cold start TrackRL (Luna) is best or tied best in all eight columns of Table 2, at 23.2% TGC-N skill-assisted and 22.6% skill-free against 15.5% for GRPO; under warm start it leads seven of the eight, at 33.3% and 34.5% TGC-N.

Ablation Study. To isolate the contributions of skill tracking and lifecycle management, we compare TrackRL with three variants on ALFWorld. **(1) w/o tracking** jointly injects or withholds retrieved skills, yielding only bundle-level utility. **(2) w/o management** admits skills but never retires,

Table 2: **Results on AppWorld.** TGC and SGC (%) on the normal (N) and challenge (C) splits in the skill-assisted and skill-free modes on the same checkpoint. Luna and Flash denote the reflection model. In each setting the best summary result is shown in **bold**.

Method	Skill-assisted				Skill-free			
	TGC-N	SGC-N	TGC-C	SGC-C	TGC-N	SGC-N	TGC-C	SGC-C
Closed-source LLMs								
Gemini-2.5-Flash	-	-	-	-	41.1	26.8	28.3	15.1
GPT-5.6-Luna	-	-	-	-	83.9	76.8	76.7	58.3
RL from the base model (Qwen3-4B-Instruct-2507)								
Origin	-	-	-	-	5.9	0.0	2.4	0.7
GRPO	-	-	-	-	15.5	7.1	3.8	0.7
SkillRL (Flash)	20.8	**10.7**	6.5	1.4	16.7	8.9	5.5	1.4
SkillRL (Luna)	20.8	7.1	7.0	2.2	20.8	5.4	7.4	2.2
D2Skill (Flash)	15.5	7.1	2.9	0.0	14.9	5.4	4.6	0.7
D2Skill (Luna)	17.3	7.1	2.6	0.0	20.8	10.7	4.1	0.0
TrackRL (Flash)	20.2	5.4	5.5	1.4	17.9	8.9	6.5	0.7
TrackRL (Luna)	**23.2**	**10.7**	**8.9**	**2.9**	**22.6**	**12.5**	**10.8**	**2.9**
RL from the SFT (Qwen3-4B-Instruct-2507 + SFT, Luna reflection)								
Origin + SFT	-	-	-	-	31.6	16.1	17.5	**7.9**
GRPO	-	-	-	-	30.4	16.1	13.2	5.8
SkillRL	30.9	12.5	18.0	7.9	32.7	16.1	17.5	**7.9**
D2Skill	26.8	12.5	10.3	2.2	25.6	10.7	12.7	5.0
TrackRL	**33.3**	**14.3**	**21.1**	**8.6**	**34.5**	**17.9**	**20.1**	**7.9**

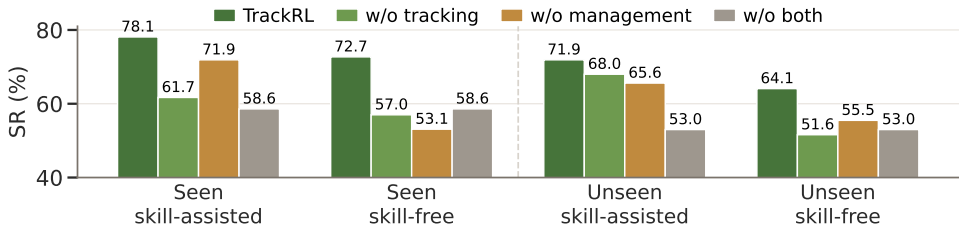


Figure 3: **Ablation on ALFWorld.** Success rate (%) of TrackRL and its three variants in the four evaluation settings, namely the seen and unseen splits evaluated in the skill-assisted and skill-free modes. The variant w/o both has no library, so its two modes coincide.

removes, or evicts them. **(3) w/o both** uses no library and reduces to plain GRPO. Figure 3 supports three conclusions. (1) Internalization requires the full mechanism. Only TrackRL improves skill-free SR over GRPO on both splits, reaching 72.7% versus 58.6% on seen tasks, while both ablated variants remain near the GRPO level. This suggests that the library strengthens the policy itself when its skills are both tracked and managed. (2) Skill tracking is critical to effective library use. Without per-skill attribution, seen skill-assisted SR falls from 78.1% to 61.7%, close to GRPO, as reranking and lifecycle decisions can no longer reliably distinguish useful skills from the rest. (3) Lifecycle management matters most for internalization. Removing all exits lowers seen skill-assisted SR from 78.1% to 71.9% but skill-free SR from 72.7% to 53.1%, suggesting that indefinitely retained skills may leave the policy reliant on external guidance.

Skill Library Growth. To compare library growth under different management schemes, we track the library sizes of SkillRL, D2Skill, and TrackRL and decompose TrackRL by skill state in Figure 4. Three patterns emerge. (1) After similar initial growth, the libraries follow distinct dynamics. SkillRL accumulates skills continuously, D2Skill fluctuates under periodic pruning, and TrackRL initially stabilizes near capacity as evidence-gated exits balance new admissions. (2) TrackRL keeps skills under evaluation and those with positive utility in the library. Cold start from a base model produces more skills in incubation due to frequent early failures, whereas warm start from an SFT-initialized model enables faster evaluation and retirement, keeping the library below capacity. (3) The soft capacity protects skills still in incubation. When no sufficiently evaluated non-beneficial skill remains eligible for eviction, TrackRL allows the library to temporarily exceed capacity. As evidence accumulates, skill library growth is controlled through informed lifecycle decisions.

Skills on the Utility–Mastery Plane. To examine how utility and mastery jointly inform a skill's fate, we plot every active task-level skill under cold start at four training steps by its current utility

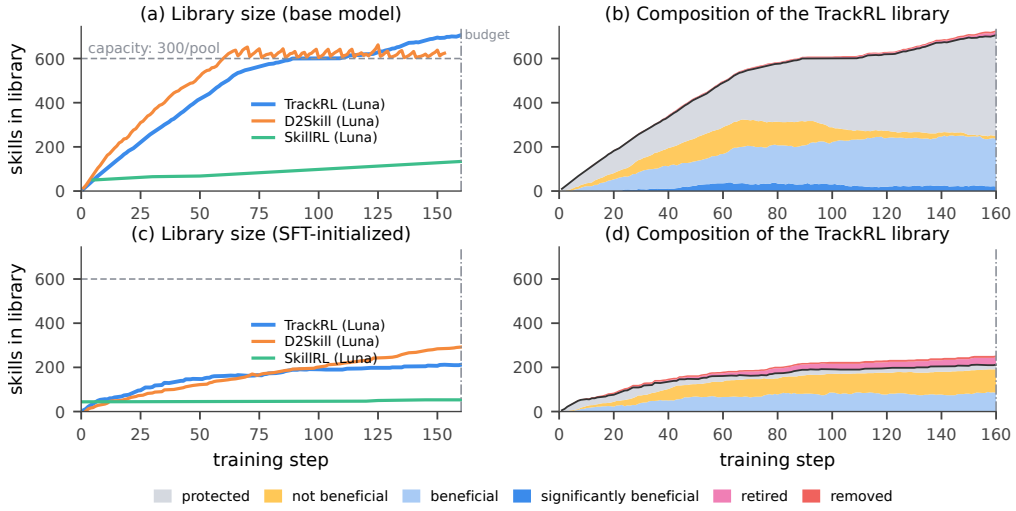


Figure 4: **Skill library growth on ALFWorld.** (a, c) Library size of TrackRL, D2Skill, and SkillRL under cold start and warm start. (b, d) Composition of the TrackRL library by skill state. Significantly beneficial skills have utility exceeding zero by two standard errors.

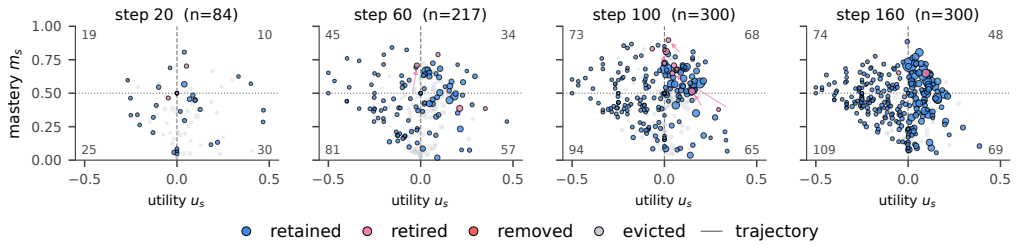


Figure 5: **Task-level skills on the utility–mastery plane.** Each marker represents an active skill positioned by its current utility u_s and mastery m_s, with color indicating its eventual lifecycle outcome and size reflecting its accumulated evidence. Corner numbers report the skill count in each quadrant, while arrows trace the earlier positions of skills later retired or removed.

and mastery, with color indicating its eventual fate, as shown in Figure 5. We have the following observations. (1) The population moves upward as the policy learns, with the share of skills whose mastery exceeds 0.5 rising from 35% at step 20 to 41% at step 160. (2) Utility estimates concentrate near zero as evidence accumulates. Extreme estimates tend to come from skills with few comparisons, whereas skills with more evidence cluster near zero, underscoring the need to consider accumulated evidence rather than a noisy point estimate. (3) Rising mastery is accompanied by declining utility. At the population level, the upper-right quadrant of positive utility and high mastery shrinks from 68 skills at step 100 to 48 at step 160, while the upper-left quadrant remains stable. At the individual level, arrows show skills later retired gaining mastery and approaching zero utility before they leave. Retirement thus appears at high mastery, retention at positive utility, and capacity eviction near zero utility, with evidence determining when a skill can leave the library.

Skill Library Lifecycle Evolution. To trace how the library evolves, we aggregate TrackRL's cold-start lifecycle events over five-step intervals in Figure 6. Early admissions drive rapid expansion, while deduplication filters repeated proposals. Retirement and harmful removal also begin once the first skills accumulate sufficient evidence, though both remain rare. After the first capacity eviction at step 67, evictions roughly balance admissions and temporarily hold the library near capacity. As protected skills accumulate and no evaluated non-beneficial skills remain eligible for eviction, the soft bound allows the library to exceed capacity rather than evict skills still in incubation. Thus, new skills continue to enter throughout training, while capacity eviction accounts for most exits and retirement and harmful removal occur selectively when supported by evidence.

Three Different Skill Lifecycles. We trace three skills that are ultimately removed, retired, and retained, respectively (Figure 7). (1) A step-level skill is protected during incubation and removed at step 91 when its utility reaches -0.11 and its upper confidence bound falls below zero. (2) A task-level skill is retired at step 39 as its mastery rises from 0.5 to 0.97 while its utility remains at

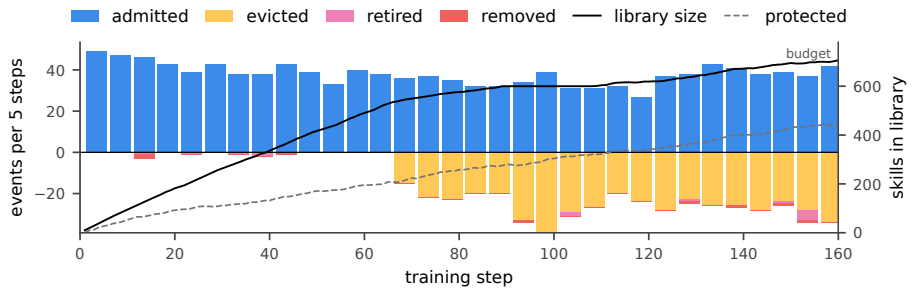


Figure 6: **TrackRL lifecycle events on ALFWorld.** Bars show admissions upward and exits downward over five-step intervals, while lines show active and protected skill counts.

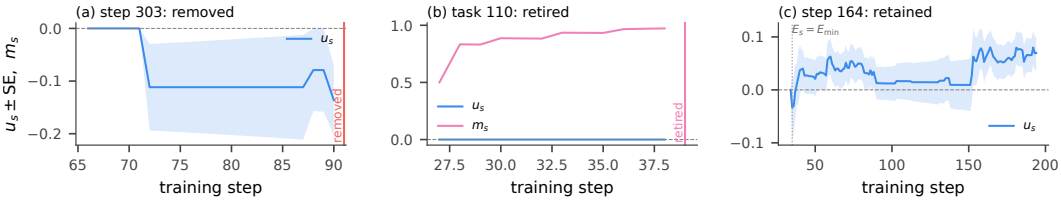


Figure 7: **Three skill lifecycles under cold start.** The three skills end in removal, retirement, and retention, respectively. Shaded bands indicate utility confidence intervals.

zero. (3) Another step-level skill remains in the library through training, with utility of 0.07 after 310 comparisons and a confidence interval above zero once estimable. Together, these cases show how skill utility, model mastery, and accumulated evidence guide distinct lifecycle decisions.

4 Related Work

Skill-Augmented Agentic RL. Recent skill-augmented agentic RL methods differ in how they use skills. Some distill reusable skills from experience and retrieve them as prompt context for subsequent rollouts and policy optimization (He et al., 2026; Zhang et al., 2026). Others adapt the strength, timing, and content of skill guidance to task outcomes or policy competence, sometimes learning when and which skill to invoke (Xiao et al., 2026; Yang et al., 2026; Shen et al., 2026). A further direction uses skills as privileged training-time context and distills the resulting behavior into a skill-free policy (Wang et al., 2026b; Lu et al., 2026; Lin et al., 2026a; Zhu et al., 2026). Despite these advances, skill evaluation remains largely based on library-level outcomes, while per-skill estimates rely on one-off validation or post-hoc attribution (Wang et al., 2026c; Zhao et al., 2026). TrackRL instead identifies each skill's contribution during training and tracks its changing value as the policy evolves, providing evidence for full lifecycle management.

Skill Library Lifecycle Management. Skill libraries require continual maintenance, as stored entries may drift, conflict with subsequent actions, or contain poisoned experience (Dabas et al., 2026; Chen et al., 2026). Existing systems rely on write-time admission (Zhang & Li, 2026), decision-time arbitration (Dong et al., 2026), pruning, merging, and refactoring (Pu et al., 2026; Lin et al., 2026b), or consolidation of recurring experience into parameters (Xia et al., 2026b). These decisions, however, focus on entry properties such as consistency, redundancy, and staleness rather than whether the evolving policy still needs each skill as its capabilities shift (Yu et al., 2026b). As a result, ineffective, internalized, and harmful skills remain difficult to distinguish, although each calls for a different lifecycle decision. TrackRL addresses this distinction through evidence-gated state transitions, using skill utility and model mastery to guide retention, retirement, and removal.

5 Conclusion

We presented TrackRL, an agentic RL framework in which skill tracking drives full lifecycle management of the skill library. By turning grouped sampling into an online randomized controlled experiment, TrackRL estimates skill utility, each skill's causal contribution as external scaffolding, and model mastery, the policy's ability to succeed without a corresponding task-level skill. These estimates guide evidence-gated skill lifecycle management from generation and admission through protected incubation to retention, retirement, or removal. TrackRL outperforms baselines in both skill-assisted and skill-free evaluation. Further analyses show how these decisions adapt the library to the evolving policy, closing the loop of policy–library co-evolution.

AI USE STATEMENT

In this work, we used generative AI tools to assist with code implementation, literature review, and editing of the manuscript text. All research ideas, method design, experimental analysis, and final claims are the authors' own, and all AI-assisted content was reviewed and verified by the authors.

REFERENCES

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent S: An open agentic framework that uses computers like a human. *arXiv preprint arXiv:2410.08164*, 2024. URL https://arxiv.org/abs/2410.08164.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002. doi: 10.1023/A:1013689704352.

Jialuo Chen, Lingqi Jiang, Xinhao Deng, Xiaohu Du, Jianan Ma, Yunhao Feng, Yuqi Qing, Zhihao Yuan, Linkang Du, and Jingyi Wang. When experience becomes instruction: Trajectory poisoning in self-evolving agent skill systems. *arXiv preprint arXiv:2608.05563*, 2026. URL https://arxiv.org/abs/2608.05563.

Mahavir Dabas, Jihyun Jeong, Ming Jin, and Ruoxi Jia. Memory-induced tool-drift in LLM agents. *arXiv preprint arXiv:2605.24941*, 2026. URL https://arxiv.org/abs/2605.24941.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit CoT to implicit CoT: Learning to internalize CoT step by step. *arXiv preprint arXiv:2405.14838*, 2024. URL https://arxiv.org/abs/2405.14838.

Jiajun Dong, Yutao Hu, Fengrui Fan, Shihan Dou, Yueming Wu, and Deqing Zou. MemArbiter: Decision-time memory arbitration for long-horizon LLM agents. *arXiv preprint arXiv:2608.02113*, 2026. URL https://arxiv.org/abs/2608.02113.

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

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. URL https://arxiv.org/abs/2606.01619.

Longtao Hu, Xiao Liang, and Linchao Zhu. From interaction traces to persistent skills: Online evolution for computer- use agents. *arXiv preprint arXiv:2609.04869*, 2026. URL https://arxiv.org/abs/2609.04869.

Qinfeng Li, Yuntai Bao, Xinyan Yu, Hongze Chen, Yanming Liu, Huifeng Zhu, Yier Jin, Jintao Chen, Wenqi Zhang, and Xuhong Zhang. AttriMem: Attribution-guided process feedback for agent memory construction. *arXiv preprint arXiv:2607.21106*, 2026a. URL https://arxiv.org/abs/2607.21106.

Xiaoyuan Li, Moxin Li, Keqin Bao, Yubo Ma, Wenjie Wang, Dayiheng Liu, and Fuli Feng. SkillGraph: Skill-augmented reinforcement learning for agents via evolving skill graphs. *arXiv preprint arXiv:2605.12039*, 2026b. URL https://arxiv.org/abs/2605.12039.

Hongxiang Lin, Zhirui Kuai, Erpeng Xue, and Lei Wang. SkillC: Learning autonomous skill internalization in LLM agents via contrastive credit assignment. *arXiv preprint arXiv:2605.27899*, 2026a. URL https://arxiv.org/abs/2605.27899.

Huawei Lin, Peng Li, Jie Song, Fuxin Jiang, and Tieying Zhang. MUSE-Autoskill: Self-evolving agents via skill creation, memory, management, and evaluation. *arXiv preprint arXiv:2605.27366*, 2026b. URL https://arxiv.org/abs/2605.27366.

Ming Liu. More is not always better: Cross-component interference in LLM agent scaffolding. *arXiv preprint arXiv:2605.05716*, 2026. URL https://arxiv.org/abs/2605.05716.

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. URL https://arxiv.org/abs/2604.02268.

Hongji Pu, Xinyuan Song, and Liang Zhao. SkillOps: Managing LLM agent skill libraries as self-maintaining software ecosystems. *arXiv preprint arXiv:2605.13716*, 2026. URL https://arxiv.org/abs/2605.13716. Submitted to NeurIPS 2026; not accepted at the time of checking.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

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

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. URL https://arxiv.org/abs/2605.06130.

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 (ICLR)*, 2021. URL https://arxiv.org/abs/2010.03768.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. *arXiv preprint arXiv:2504.07952*, 2025. URL https://arxiv.org/abs/2504.07952.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. URL https://arxiv.org/abs/2407.18901.

Songjun Tu, Chengdong Xu, Qichao Zhang, Yaocheng Zhang, Xiangyuan Lan, Linjing Li, Dong Li, and Dongbin Zhao. Dynamic dual-granularity skill bank for agentic RL. *arXiv preprint arXiv:2603.28716*, 2026. URL https://arxiv.org/abs/2603.28716.

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

Hanyang Wang, Weijieying Ren, Yuxiang Zhang, Ding Cao, Zhizhao Zeng, Ke Zeng, and Tianxiang Zhao. BiPACE: Bisimulation-guided policy optimization with action counterfactual estimation for LLM agents. *arXiv preprint arXiv:2606.25556*, 2026a. URL https://arxiv.org/abs/2606.25556.

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*, 2026b. URL https://arxiv.org/abs/2604.10674.

Yixuan Wang, Yiyang Zhou, Yiming Liang, Congyu Zhang, Fuxiao Liu, Jiawei Zhou, and Huaxiu Yao. Not all skills help: Measuring and repairing agent knowledge. *arXiv preprint arXiv:2606.15390*, 2026c. URL https://arxiv.org/abs/2606.15390.

Zhenhailong Wang, Haiyang Xu, Junyang Wang, Xi Zhang, Ming Yan, Ji Zhang, Fei Huang, and Heng Ji. Mobile-agent-E: Self-evolving mobile assistant for complex tasks. *arXiv preprint arXiv:2501.11733*, 2025. URL https://arxiv.org/abs/2501.11733.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024. URL https://arxiv.org/abs/2409.07429.

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*, 2026a. URL https://arxiv.org/abs/2602.08234.

Siyu Xia, Chenheng Zhang, Yanting Wu, Haoxuan Li, Jiajun Chai, Xiaohan Wang, Guojun Yin, Wei Lin, Zhouchen Lin, Haifeng Zhang, and Jun Wang. UniMem: Complementary episodic-to-parametric memory for boundary-agnostic task streams. *arXiv preprint arXiv:2607.26017*, 2026b. URL https://arxiv.org/abs/2607.26017.

Han Xiao, Guozhi Wang, Hao Wang, Shilong Liu, Yuxiang Chai, Yue Pan, Yufeng Zhou, Xiaoxin Chen, Yafei Wen, and Hongsheng Li. UI-Mem: Self-evolving experience memory for online reinforcement learning in mobile GUI agents. *arXiv preprint arXiv:2602.05832*, 2026. URL https://arxiv.org/abs/2602.05832.

Shidong Yang, Ziyu Ma, Tongwen Huang, Xucong Wang, Renda Li, Yiming Hu, Yong Wang, and Xiangxiang Chu. SkillForge: Evolving verifiable skills for reinforcement learning agents. *arXiv preprint arXiv:2608.24747*, 2026. URL https://arxiv.org/abs/2608.24747.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2207.01206.

Zhiyuan Yao, Yuxin Chen, Zhengxi Lu, Zishan Xu, Yueqing Sun, Yifu Guo, Yuquan Lu, Zhengzhou Cai, Kangning Zhang, Zhuowen Han, Zi-Han Wang, Ziang Ye, Qi Gu, Xunliang Cai, Weiwen Liu, and Yongliang Shen. SkillRise: Agentic reinforcement learning for cross-task skill evolution. *arXiv preprint arXiv:2607.26784*, 2026. URL https://arxiv.org/abs/2607.26784.

Jianxiang Yu, Jiapeng Zhu, Bochen Lin, Qier Cui, Zichen Ding, and Xiang Li. Skill is not one-size-fits-all: Model-aware skill alignment for LLM agents. *arXiv preprint arXiv:2605.30723*, 2026a. URL https://arxiv.org/abs/2605.30723.

Ye Yu, Xiaopeng Yuan, Haibo Jin, Heming Liu, Yaoning Yu, and Haohan Wang. Do self-evolving agents forget? Capability degradation and preservation in lifelong LLM agent adaptation. *arXiv preprint arXiv:2605.09315*, 2026b. URL https://arxiv.org/abs/2605.09315.

Yan Zhang and Shibo Li. ConsistencyGate: Preventing memory contamination in LLM agents via self- consistency admission control. *arXiv preprint arXiv:2607.22962*, 2026. URL https://arxiv.org/abs/2607.22962.

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. URL https://arxiv.org/abs/2606.08755.

Qiyan Zhao, Xiaofeng Zhang, Bo Liu, Minda Chen, Wei Xiong, Jingyang Chen, Guanting Ye, Wenhao Yu, Xiaosong Yuan, Shijie Han, Da-Han Wang, Jianmin Ji, Fei Huang, and Xu-Yao Zhang. Coalition-aware skill reliability for self-evolving agents. *arXiv preprint arXiv:2608.22610*, 2026. URL https://arxiv.org/abs/2608.22610.

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. URL https://arxiv.org/abs/2605.28424.

CONTENTS

1 Introduction **1**

2 TrackRL **3**

2.1 Skill-Augmented RL with Skill Tracking 3

2.2 Skill Library Full Lifecycle Management 5

3 Experiments **6**

4 Related Work **9**

5 Conclusion **9**

A Benchmarks **14**

B Baselines **14**

C Evaluation **15**

D Implementation Details **16**

D.1 Skills Organization, Retrieval, and Injection 16

D.2 Training Procedure 16

D.3 Hyperparameters 19

D.4 Computational Resources 19

E Prompt Templates **20**

E.1 Skill Generation and Reflection 20

E.2 Retrieval and Injection Format 21

F Additional Results **21**

F.1 Per-Class Breakdown on ALFWorld 21

F.2 Internalization by Task Family 21

F.3 Library Size for the SFT-Initialized Policy 22

G Broader Impacts **23**

Appendix

The appendix provides the experimental details, implementation, prompts, and supplementary results referenced in the main text. Appendix A describes the two benchmarks, their splits, and the metrics we report. Appendix B describes each baseline and the settings shared across methods. Appendix C gives the evaluation protocol, including the two evaluation modes, the evaluation prompts, and the episode budget. Appendix D gives the implementation of TrackRL, covering the skill record with an example of each granularity, the skill pools and retrieval, the full training step as an algorithm, the hyperparameters, and the computational cost. Appendix E lists the prompt used for skill generation and the format in which skills are injected. Appendix F reports additional results, namely the per-class breakdown on the unseen split of ALFWorld, internalization by task family, and the library under warm start. Appendix G discusses the broader impacts of this work.

A Benchmarks

We evaluate on two benchmarks of different form, a text-based household environment and a code-based environment of applications and APIs.

ALFWorld (Shridhar et al., 2021) is a text-based household environment with six task families, namely Pick, Clean, Cool, Look, Heat, and Pick2. We train on the official training games and evaluate on the *seen* split of 140 games and on the *unseen* split of 134 games in room layouts absent from training. We follow the standard protocol of RL-trained ALFWorld agents (Feng et al., 2025; Xia et al., 2026a), with 128 games sampled per split, sampling temperature 0.4, at most 50 turns, and a frozen library during evaluation, and we report the success rate (SR).

AppWorld (Trivedi et al., 2024) is a code-based environment of 9 applications and 457 APIs in which the agent completes tasks by writing API calls. We train on the train split and evaluate on *test-normal* with 168 tasks drawn from 56 scenarios and on *test-challenge* with 417 tasks drawn from 139 scenarios, using the official ReAct prompt, at most 40 turns, and greedy decoding. A scenario is a single situation instantiated as three related task variants, so both splits contain exactly three tasks per scenario. We report task goal completion (TGC), which counts a task as solved when all of its unit tests pass, and scenario goal completion (SGC), which counts a scenario as solved only when all three of its tasks are solved, making SGC the stricter of the two metrics.

B Baselines

We compare TrackRL against the three groups of methods used in Section 3. Every method sees the same environments, the same interaction prompt, and the same turn cap, and every trained method additionally shares the policy model, the group size, and the batch size, so that a difference between two rows is attributable to the method rather than to its setting.

Proprietary LLMs. Two closed-source models under direct prompting, with no training and no skill library, are included to show what a base model achieves on these tasks rather than as trained competitors.

- **Gemini-2.5-Flash** is prompted directly on the same tasks, with the same interaction prompt and the same turn cap as every trained method.
- **GPT-5.6-Luna** follows the same direct-prompting protocol. Both models also serve as reflection models for the skill-augmented methods, so the comparison also shows what the model generating the skills can do on the task itself.

Skill-free Models. Evaluating the policy without a skill library measures what it can accomplish using its learned capabilities alone. This distinguishes improvements retained by the policy from gains that depend on retrieving skills as external support at evaluation time.

- **Origin** is the untrained base model, Qwen3-4B-Instruct-2507, prompted with the same interaction format and given no library. It provides a common reference for measuring gains from training.

- **SFT** is the base model after supervised fine-tuning on demonstration trajectories, with no reinforcement learning and no library. The SFT model appears in our tables both as a standalone baseline and as the second initialization for every RL method.

- **GRPO** (Shao et al., 2024) is group relative policy optimization, which removes the value network and standardizes the reward of each episode against the other episodes of its group, so that the group itself supplies the baseline. GRPO underlies TrackRL's policy optimization but uses no skill mechanism, so its performance gap with TrackRL reflects the contribution of the skill library under the same training objective.

Skill-augmented RL methods. These methods build a skill library during training and inject retrieved skills into the prompt, as TrackRL does. Both add an intrinsic reward that favors skill-conditioned over skill-free episodes. We omit it in our runs to avoid rewarding reliance on the skill library as external scaffolding rather than task completion.

- **SkillRL** (Xia et al., 2026a) is recursive skill-augmented reinforcement learning, in which the agent distills skills from its own trajectories and reuses them in later episodes as the library accumulates. Skills enter the library and are never removed or retired, so SkillRL is our reference point for what unmanaged accumulation costs over a long run.

- **D2Skill** (Tu et al., 2026) is a dynamic dual-granularity skill bank that maintains task-level and step-level skills, the same two granularities we adopt, and updates them during training. It is the closest method to ours and the most informative comparison, because it differs mainly in what it measures. D2Skill scores the retrieved bundle rather than the individual skills inside it, so a skill can inherit credit from whatever was retrieved alongside it, which is the free-riding that per-skill randomization is designed to remove.

We train every skill-augmented RL method from both the base model and the SFT checkpoint. We also run every skill-augmented method with each of two reflection models, GPT-5.6-Luna and Gemini-2.5-Flash, so that the findings do not depend on a single reflection model. All skill-augmented methods use the same retrieval embedding model, holding this component fixed when comparing how they manage their libraries.

C EVALUATION

Evaluation modes. Each checkpoint is evaluated in two modes on the same tasks. In the *skill-assisted* mode, skills are retrieved and injected exactly as in training, which measures the agent and its library as a whole and matches the reporting of prior work. In the *skill-free* mode, nothing is retrieved and the prompt is identical to that of plain GRPO, which measures the capability that has entered the policy parameters. The difference between the two modes on the same checkpoint is the net effect of the library on that policy, and for a method without a library the two modes coincide.

Evaluation prompts. Evaluation uses the same interaction prompt as training. On ALFWorld this is the standard ReAct-style prompt of the environment, with the injected block inserted exactly as in Appendix E.2, and on AppWorld it is the official ReAct prompt distributed with the benchmark. Two things differ from training. The library is frozen, so no skill is admitted, retired, or removed during an evaluation pass, and nothing is randomized, so every selected skill is injected in the skill-assisted mode and the block is absent in the skill-free mode. The skill-free prompt is therefore identical to the prompt of plain GRPO, which is what makes the two modes comparable across methods with and without a library.

Episode budget. An episode ends when the task is solved, when the environment declares failure, or when the turn cap is reached. The cap is 50 turns on ALFWorld and 40 turns on AppWorld, and it is the same for every method, including the proprietary models under direct prompting. The cap matters for the reported numbers because an unsolved episode consumes the whole budget. Across the ALFWorld validation passes, failed episodes average close to the 50-turn cap while solved episodes finish in well under half of it, so a method with a lower success rate also consumes more interaction per solved task.

D Implementation Details

D.1 Skills Organization, Retrieval, and Injection

Skill record. A skill is a short textual record with three fields. The *title* is a few words naming the skill and appears as the heading of its entry in the injected context. The *principle* is one or two sentences stating what to do and carries the substantive content of the skill. The *when-to-apply* clause is one sentence describing the situation the skill addresses. This clause serves as the retrieval key $\kappa(s)$ in Eq. (1), so a skill is matched by the situation it claims to cover rather than by the wording of its advice. Alongside its text, a skill stores its tracked statistics, its exposure count N_s, and the step at which it was admitted. A task-level skill additionally records the task family of the trajectory from which it was distilled, which supplies $f(s)$ in Eq. (12).

Skill Pools. The two pools differ in when they are queried and how long injected skills remain in context. The task-level pool $\mathcal{P}_{\text{task}}$ is queried once at the beginning of each episode using the task description. When injected, a task-level skill appears before the first observation and remains in context throughout the episode. Retrieval is restricted to the episode's task family, with no backfilling from other families. The step-level pool $\mathcal{P}_{\text{step}}$ is queried at every turn using the task description concatenated with the current observation. When injected, a step-level skill follows that observation and guides only the next action. Both pools start empty and are populated solely through reflection on failures during training.

The example below shows a task-level and a step-level skill generated by the reflection model for the heating family of ALFWorld, written as the JSON records returned by reflection (Appendix E.1).

```
Task-level skill
{
  "title": "Complete the heating-and-storage sequence",
  "principle": "Locate the potato, place it in the microwave,
    activate the microwave to heat it, then remove it and put it
    in the fridge; do not confuse opening or closing the appliance
    with heating.",
  "when_to_apply": "When the task requires heating an object and
    subsequently storing it in a different receptacle"
}

Step-level skill
{
  "title": "Search likely cooking locations systematically",
  "principle": "After checking the fridge, inspect the stove or
    other likely egg locations before wandering through unrelated
    containers.",
  "when_to_apply": "When the target object has not been found and
    the task involves heating or cooking it"
}
```

Retrieval and Assignment. We compare queries and retrieval keys by the cosine similarity of their embeddings. If no key exceeds the retrieval threshold, we return an empty set rather than a weak match. Exposure counts used in the reranking bonus accumulate without discounting. We draw each task-level assignment once at episode onset and each step-level assignment upon first retrieval, reusing it whenever the skill is retrieved again in the same episode. In the two skill-free episodes reserved per group, retrieval and reranking remain unchanged, but the corresponding assignments are fixed to zero, so their outcomes enter the control arm for every skill selected there. Injected skills appear in up to two short Markdown sections, one per granularity (Appendix E.2). When no skill is injected, we omit the entire block, leaving the same prompt as in plain GRPO.

D.2 Training Procedure

This section states one full training step of TrackRL with every equation of Section 2.1 and Section 2.2 written out in the place where it is applied. Table 3 collects the notation, and Algorithm 1 gives the procedure.

Table 3: Mathematical Notation. The upper block is the sampling and optimization setting, the middle block is the state TrackRL tracks for a skill, and the lower block is the parameters that govern retrieval and the lifecycle.

Symbol	Meaning	Introduced in
$\pi_\theta,\ \pi_{\text{old}},\ \pi_{\text{ref}}$	current, behavior, and reference policy	Eq. (3)
$M,\ B$	tasks per step and episodes per group	Section 2.1
$g,\ e$	a group of episodes and a single episode	Section 2.1
$x,\ \kappa(s)$	retrieval query and the retrieval key of skill s	Eq. (1)
$y_e,\ \phi_e,\ R_e,\ A_e$	outcome, invalid-action fraction, reward, group-normalized advantage	Eq. (3)
$z_{s,e}$	injection assignment of skill s in episode e	Eq. (4)
$\mathcal{P}_{\text{task}},\ \mathcal{P}_{\text{step}},\ \mathcal{H}$	task-level pool, step-level pool, record of harmful skills	Section 2.2
$n^+_{s,g},\ n^-_{s,g}$	treatment and control arm sizes of s in group g	Eq. (5)
$c^+_{s,g},\ c^-_{s,g}$	successes in the two arms	Eq. (5)
$d_{s,g},\ w_{s,g}$	paired comparison and its reliability weight	Eq. (5)
$W_s,\ S_s,\ Q_s,\ G_s$	discounted running statistics of s	Eq. (6)
$u_s,\ \text{SE}(u_s)$	utility estimate and its standard error	Eq. (7)
$n^-_s,\ c^-_s$	discounted control-arm episode and success counts	Eq. (8)
$m_s,\ \text{SE}(m_s)$	mastery estimate and its standard error	Eq. (9)
E_s	undiscounted evidence count since admission	Section 2.1
$N_s,\ N_{\mathcal{P}}$	exposure count of s and of its pool	Eq. (2)
$b_{f(s)}$	skill-free success rate of the family of s	Eq. (12)
$\delta,\ m,\ K$	similarity threshold, candidates kept, skills selected	Eq. (1)
$\alpha,\ \eta$	relevance-evidence balance and exploration coefficient	Eq. (2)
p	injection probability in randomized episodes	Eq. (4)
γ	recency discount	Eq. (6)
$\lambda,\ E_{\min},\ m_{\min}$	confidence level, evidence threshold, mastery floor	Eq. (11), Eq. (12)
$\delta_{\text{dup}},\ A,\ C$	deduplication threshold, admission cap, soft capacity	Eq. (10), Section 2.2
$\epsilon,\ \beta,\ \nu$	clipping range, KL weight, invalid-action penalty	Eq. (3)

Algorithm 1 One training step of TrackRL. The library is frozen for the duration of the step.

Require: π_θ, pools $\mathcal{P}_{\text{task}}, \mathcal{P}_{\text{step}}$, $\mathcal{H}$, per-skill state, family baselines b_f

Stage 1. Sampling. Sample M tasks and roll out one group g of B episodes each under π_{old}.
for each group g **do**
  Reserve one task-level and one step-level skill-free episode, which use $p = 0$.
  for each episode $e \in g$, per episode for task-level and per turn for step-level **do**
    $\mathcal{R}(x; \mathcal{P}) = \text{Top}_m \{ s \in \mathcal{P} \mid \text{sim}(x, \kappa(s)) \geq \delta \}$ Eq. (1)
    $\text{score}(s) = \alpha \widetilde{\text{sim}}(x, \kappa(s)) + (1-\alpha)\big[u_s + \eta\sqrt{\log(1+N_{\mathcal{P}})/(1+N_s)}\big]$ Eq. (2)
    Take the top K, draw $z_{s,e} \sim \text{Bernoulli}(p)$ independently per skill Eq. (4)
    Inject the skills with $z_{s,e} = 1$, act, and increment N_s and $N_{\mathcal{P}}$
  end for
  Observe y_e and record each episode with its offered skills and assignments
end for
Stage 2. Tracking. Let $\mathcal{G}_s$ be the groups of this step in which both arms of s are non-empty.
for each skill s and each $g \in \mathcal{G}_s$ **do**
  $d_{s,g} = \frac{c^+_{s,g}}{n^+_{s,g}} - \frac{c^-_{s,g}}{n^-_{s,g}}, \quad w_{s,g} = \frac{n^+_{s,g} n^-_{s,g}}{n^+_{s,g} + n^-_{s,g}}$ Eq. (5)
end for
for each skill s in either pool **do**
  $W_s \leftarrow \gamma W_s + \sum_g w_{s,g}, \quad S_s \leftarrow \gamma S_s + \sum_g w_{s,g} d_{s,g}, \quad Q_s \leftarrow \gamma Q_s + \sum_g w_{s,g} d^2_{s,g}, \quad G_s \leftarrow \gamma G_s + |\mathcal{G}_s|$ Eq. (6)
  $u_s = S_s/W_s, \quad \text{SE}(u_s) = \sqrt{(Q_s - S_s^2/W_s)/(W_s(G_s-1))}$ for $G_s \geq 2, \quad E_s \leftarrow E_s + |\mathcal{G}_s|$ undiscounted Eq. (7)
  if s is task-level **then**
    $n^-_s \leftarrow \gamma n^-_s + \sum_g n^-_{s,g}, \quad c^-_s \leftarrow \gamma c^-_s + \sum_g c^-_{s,g}$ Eq. (8)
    $m_s = \frac{c^-_s + 1}{n^-_s + 2}, \quad \text{SE}(m_s) = \sqrt{m_s(1-m_s)/(n^-_s + 3)}$ Eq. (9)
  end if
end for
Update each family baseline b_f by the same recursion over the skill-free episodes of family f
Stage 3. Lifecycle.
for each skill s in either pool **do**
  if $E_s \geq E_{\min}$ **and** $u_s + \lambda \,\text{SE}(u_s) < 0$ **then**
    Remove s, $\quad \mathcal{H} \leftarrow \mathcal{H} \cup \{s\}$ Eq. (11)
  else if s task-level **and** $E_s \geq E_{\min}$ **and** $u_s \leq 0$ **and** $m_s - \lambda \,\text{SE}(m_s) \geq \max\{b_{f(s)}, m_{\min}\}$ **then**
    Retire s Eq. (12)
  end if
end for
for each candidate from reflection on the lowest-scoring groups, at most A per pool **do**
  if $\max_{s \in \mathcal{P} \cup \mathcal{H}} \text{sim}(s_{\text{new}}, s) < \delta_{\text{dup}}$ **then**
    Admit s_{new} with empty statistics Eq. (10)
  end if
end for
while $|\mathcal{P}| > C$ **and** $\{s : E_s \geq E_{\min} \wedge u_s \leq 0\} \neq \emptyset$ **do**
  Evict the eligible skill of smallest u_s
end while
Stage 4. Policy update. $R_e = y_e - \nu\phi_e, \quad A_e = (R_e - \text{mean}_{e' \in g} R_{e'}) / \text{std}_{e' \in g} R_{e'}$
$\max_\theta \; \mathcal{J}(\theta) = \mathbb{E}_g\big[\frac{1}{B}\sum_{e \in g} \frac{1}{L_e} \sum_k \min(\rho_{e,k} A_e, \text{clip}(\rho_{e,k}, 1 \pm \epsilon) A_e)\big] - \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})$ Eq. (3)

D.3 HYPERPARAMETERS

Table 4 lists the reinforcement learning hyperparameters, which are shared by every trained method so that the comparison isolates the library rather than the optimizer. Table 5 lists the parameters of TrackRL together with the symbol each one carries in Section 2.1 and Section 2.2.

Table 4: Reinforcement learning hyperparameters, shared by all trained methods.

Parameter	Symbol	Value
Policy model	π_θ	Qwen3-4B-Instruct-2507
Tasks per training step	M	16
Episodes per group	B	8
Episodes per training step	$M \cdot B$	128
Training steps	—	160
Learning rate	—	1×10^{-6}
Mini-batch size (policy update)	—	128
Clipping range	ϵ	0.2
KL regularization weight	β	0.01
Invalid-action penalty	ν	1.0
Maximum prompt length	—	4096 tokens
Maximum response length per turn	—	512 tokens
Rollout engine	—	vLLM
Validation games per evaluation	—	128
Validation sampling temperature (ALFWorld)	—	0.4
Maximum turns (ALFWorld / AppWorld)	—	50 / 40

Table 5: TrackRL parameters. The upper block is introduced by skill tracking and lifecycle management. The lower block is the retrieval configuration shared with the skill-augmented baselines.

Parameter	Symbol	Value
Injection probability, randomized episodes	p	$1/2$
Skill-free episodes reserved per group	—	1 task-level, 1 step-level
Recency discount	γ	0.98
Evidence threshold for any exit	$E_{\min}$	10
Confidence level of both one-sided tests	λ	2
Absolute mastery floor for retirement	$m_{\min}$	0.5
Soft capacity per pool	C	300
Admission cap per pool per step	A	5
Deduplication threshold	δ_{dup}	0.90
Exploration coefficient in reranking	η	0.2
Relevance-evidence balance in reranking	α	0.1
Embedding model	—	Qwen3-Embedding-0.6B
Retrieval similarity threshold	δ	0.70
Candidates retrieved per pool	m	10
Skills selected per pool	K	3
Skill pools	$\mathcal{P}_{\text{task}}, \mathcal{P}_{\text{step}}$	task-level, step-level
Reflection model	—	GPT-5.6-Luna or Gemini-2.5-Flash

D.4 COMPUTATIONAL RESOURCES

Each run trains a 4B policy on a single node of NVIDIA B300 GPUs, with vLLM serving the rollouts. On ALFWorld, for example, a training step collects 128 episodes of at most 50 turns, and each turn requires one generation call from the policy, so a step performs up to 6,400 such calls. Its wall-clock cost is therefore dominated by autoregressive generation rather than by the environment or the policy update. The turns of an episode are sequential, so these calls form up to fifty successive rounds of vLLM inference, each batched across the episodes of the step. The ALFWorld simulator is text-based and steps in negligible time, and the gradient update accounts for a small share of the step. Validation runs every five steps in both evaluation modes and on both splits. A single validation pass over 128 ALFWorld games costs roughly as much as a training step,

making it the largest fixed overhead of the protocol. Every method incurs this cost. The components TrackRL adds are computationally inexpensive. Tracking uses the episode outcomes GRPO already collects and updates a constant number of scalars per skill per step. Lifecycle management performs threshold comparisons over the pool and computes one embedding per candidate for the admission test. Retrieval makes one embedding call per query and reranks at most ten candidates. Reflection makes one call to an external model per failed trajectory selected at that step. This cost is not negligible, but it is shared with the skill-augmented baselines. TrackRL adds no rollouts. The treatment and control arms of Eq. (4) are formed from the B episodes GRPO already samples, so a TrackRL step and a GRPO step collect the same number of episodes.

E Prompt Templates

E.1 Skill Generation and Reflection

At each training step, the groups with the lowest success rates supply their failed trajectories to the reflection model. Each call receives one failed trajectory, and when the same group contains a solved episode of the same task, that successful trajectory is supplied as a reference. The model is asked for three outputs. The *first error step* identifies the turn at which the failed trajectory first went wrong and anchors the step-level skill. The *step reflection* is one step-level skill addressing that turn, and the *task reflection* is one task-level skill for the task as a whole. The two skills are returned as JSON objects with exactly the three fields of Appendix D.1, allowing one record to serve as both the retrieval key and the injected text.

```
You are given one failed trajectory and one successful trajectory for
the same task. Analyze them and produce exactly three outputs.

Task: {task description}
Task Type: {task family}

Failed trajectory:
{turn-by-turn observations and actions}

Successful trajectory (reference):
{turn-by-turn observations and actions}

Output the following in order (use the exact section headers):

1) FIRST_ERROR_STEP: N
   Where N is the 1-based turn number in the *failed* trajectory where
   the agent first went wrong (e.g. Turn 1, Turn 2). Use 0 if unclear.

2) STEP_REFLECTION (one step-level experience for that step only):
   Output a JSON object with exactly: "title", "principle",
   "when_to_apply". This should be a concise experience for what to do
   at that specific step/situation (e.g. "At this step you should
   ...").

3) TASK_REFLECTION (one task-level skill for the whole task):
   Output a JSON object with exactly: "title", "principle",
   "when_to_apply". This summarizes for the *whole task*: what to
   avoid and how to succeed in this kind of task (will be retrieved by
   task description).

Output format (use these exact labels):
FIRST_ERROR_STEP: N

STEP_REFLECTION:
<single JSON object>

TASK_REFLECTION:
<single JSON object>
```

When no successful trajectory is available for the task, the first paragraph and the reference trajectory are omitted and the model is given the failed trajectory alone. A candidate whose JSON cannot be parsed is discarded. Surviving candidates enter the admission test of Eq. (10), and at most $A = 5$ per pool are admitted per step.

E.2 Retrieval and Injection Format

The skills with $z_{s,e} = 1$ are rendered in up to two Markdown sections and placed in the prompt as described in Appendix D.1. Only the title and principle are shown. The when-to-apply clause serves as the retrieval key and is not injected, so the context the policy reads states what to do rather than restating the situation it already observes. The example below uses skills from the cold-start library at step 160.

```
### Task-level experience (for this kind of task)
- **Complete required object-state changes before placement**: For
  tasks requiring a clean object in a receptacle, locate and take the
  object, clean it using an available sink, then place it in the
  target cabinet.

### Step-level experience (relevant to current situation)
- **Search likely surfaces systematically**: After opening the fridge
  and not finding the target, inspect nearby cabinets and countertops
  rather than going to an irrelevant dining table.
```

A section is omitted when no skill of that granularity is injected, and the whole block is omitted when neither pool contributes. The latter is the case in the reserved skill-free episodes and at the start of training when both pools are still empty. The task-level section is written once before the first observation and stays in context for the episode, whereas the step-level section is rewritten after each observation.

F Additional Results

F.1 Per-Class Breakdown on ALFWorld

Table 1 gives the per-class results on the seen split. Table F.1 gives the corresponding breakdown on the unseen split, for the runs we trained ourselves and whose per-episode validation records we retain.

Method	**Mode**	Pick	Clean	Cool	Look	Heat	Pick2	**Overall**
D2Skill (Luna)	skill-assisted	46.7	63.0	47.8	45.5	46.7	72.7	54.7
	skill-free	55.9	40.0	70.8	54.5	35.7	45.0	52.3
TrackRL (Luna)	skill-assisted	61.3	65.2	82.6	61.5	92.9	75.0	71.9
	skill-free	63.3	66.7	52.2	91.7	75.0	52.2	64.1
TrackRL (Luna) + SFT	skill-assisted	100.0	96.2	96.0	100.0	100.0	95.2	97.7
	skill-free	87.5	88.0	100.0	100.0	100.0	100.0	95.3

Table 6: **Results on the unseen split of ALFWorld by task class.** SR (%) by task class, with the overall values of Table 1.

Under SFT initialization the two modes nearly coincide on the unseen split, at 97.7% skill-assisted against 95.3% skill-free. Once the policy is sufficiently strong the library neither adds to nor subtracts from its unseen performance, which is the regime in which retirement rather than retention is the common outcome.

F.2 Internalization by Task Family

Figure 8 tracks three quantities for each of the six ALFWorld task families: the family utility of its task-level skills, their mean mastery, and the policy's success rate without the library. Family utility



Figure 8: **Task-level skills by family (ALFWorld, base model, Luna reflection).** One panel per family, showing the mean mastery of the family's active task-level skills on the left axis, the success rate of the policy without the library on the seen validation games of that family on the same axis as a dashed line and as a three-point moving average since each family has about twenty games per validation, the family utility pooled over the family's skills on the right axis, and one vertical tick per retirement of a skill of that family. Panel titles give the number of active task-level skills of the family at the last step and the number retired.

pools the within-group utility estimates of Section 2.1 across comparisons of the family's skills, measuring what those skills add to an episode from that family at each step. The three quantities evolve as the lifecycle framework anticipates, with the six families showing three distinct patterns.

Internalized families. In Pick, the policy solves 81% of validation games without skills by the end of training, up from 66% at the start. Family utility falls from 0.09 to 0.04, and three skills are retired. In Cool, utility peaks at 0.05 around steps 60 to 120, when mastery is highest. Five skills are retired between steps 115 and 160, and utility returns to zero by the end.

Families that continue to require support. Clean maintains a family utility of 0.10 throughout training, with skill-free success between 21% and 32%. Heat and Look end at 0.09 and 0.06, respectively. Where the policy continues to fail without skills, family utility persists.

The family the library does not affect. In Pick2, the hardest family, utility remains at zero, mastery stays at 0.33, and no skill is retired. The policy improves from 16% to 39% through reinforcement learning alone, while the library neither improves nor degrades its performance.

F.3 Library Size for the SFT-Initialized Policy

The bottom row of Figure 4 tracks the library of the SFT-initialized policy. Its trajectory differs from the cold-start run more in magnitude than in shape. The library grows rapidly to 135 skills in the first 40 steps, then levels off, reaching 212 skills at step 160, including 80 task-level and 132 step-level skills. Neither pool approaches the soft capacity of $C = 300$, so the capacity rule never binds and every exit in this run is evidence-gated rather than capacity-based.

The library remains small because skills are generated only from failures. The SFT-initialized policy already solves most training tasks, so few groups qualify among the lowest-scoring groups at each step, few candidates are proposed, and the admission cap of $A = 5$ per pool rarely binds. The composition of exits also differs. Under SFT initialization, 33 task-level skills are retired and 9 skills are deleted as harmful or duplicated by step 160. Retirement thus accounts for most exits and covers roughly a third of all task-level skills that were ever active. Under base-model initialization, the proportions are reversed, with 14 retirements against 497 deletions, and retirement covers about 3% of task-level skills that were ever active. The same lifecycle rules therefore produce a small, heavily internalized library under a strong initialization and a large, capacity-bound library under a weak one, without changing the thresholds between runs.

G BROADER IMPACTS

The components this work adds are a bookkeeping layer over an existing training pipeline, and their main external effect is to make an agent's accumulated context trackable. Every estimate and every lifecycle transition is recorded together with the statistics that justified them, so a third party can replay offline why a skill was kept, retired, or removed. We regard this as the useful direction of the contribution, since a skill library is otherwise an opaque store of model-generated text that silently shapes an agent's behavior.

The same mechanism carries the corresponding risk. Skills are distilled by a language model from its own failures, so they can encode mistaken or unsafe generalizations, and a skill that is merely useless rather than harmful is retained as long as its measured utility is not negative. The evidence threshold delays every exit by design, so a harmful skill remains in the library until enough comparisons accumulate. Deployment beyond simulated benchmarks would need the removal rule to be supplemented by checks that do not wait for statistical evidence. We also note that skill text is derived from trajectories, so in any application where trajectories contain sensitive data, the library inherits that sensitivity and should be handled as derived data rather than as configuration.