DYNAMIC SKILL LIFECYCLE MANAGEMENT FOR AGENTIC REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model agents increasingly rely on external skills to solve complex tasks, where skills act as modular units that extend their capabilities beyond what parametric memory alone supports. Existing skill-based RL methods assume external skills either accumulate as persistent guidance or internalized into the policy, eventually leading to zero-skill inference. We argue this assumption is overly restrictive, since with limited parametric capacity and uneven marginal contribution across skills, the optimal active skill set is non-monotonic, task- and stage-dependent. In this work, we propose SLIM, a framework of dynamic **S**kill **LI**fecycle **M**anagement for agentic reinforcement learning (RL), which dynamically updates the active external skill set as the policy evolves. Specifically, SLIM estimates each active skill's marginal external contribution through leave-one-skill-out validation, then applies three lifecycle operations: retaining high-value skills, retiring skills whose contribution becomes negligible after sufficient exposure, and expanding the skill bank when persistent failures reveal missing capability coverage. Experiments show that SLIM outperforms the best baselines by an average of 7.1% points across ALFWorld and SearchQA. Results further indicate that policy learning and external skill retention are not mutually exclusive: some skills are absorbed into the policy, while others continue to provide external value, supporting SLIM as a more general paradigm for skill-based agentic RL.

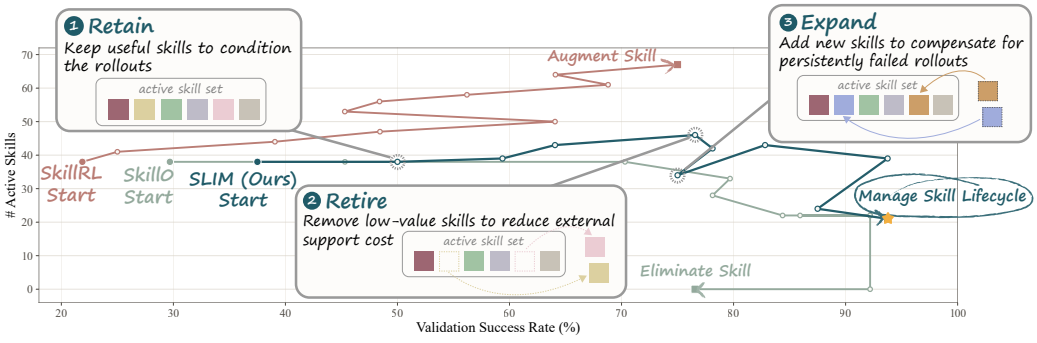


Figure 1: The reinforcement learning dynamics on ALFWorld. We plot validation success rate against the number of skills in active set during training. SkillRL accumulates external skills, whereas Skill0 progressively eliminates them. SLIM instead performs retain–retire–expand lifecycle management, converging to a non-empty skill set with higher validation success. This suggests that the effective endpoint is a learned external skill boundary rather than full accumulation or forced elimination.

1 INTRODUCTION

Large language model (LLM) agents (Luo et al., 2025; Wang et al., 2024a) are increasingly used to solve complex tasks that require multi-step reasoning (Plaat et al., 2026), long-horizon planning (Huang et al., 2024), and reliable tool use (Shen, 2024). A growing way to improve these agents is to equip them with external skills (Chen et al., 2026; Zheng et al., 2025; Ni et al., 2026), where each skill is a modular procedural artifact inserted at inference time to provide reusable task-solving

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.

guidance. (Zhou et al., 2026). By conditioning the agent on such external procedural knowledge, skill-based agents can extend capabilities beyond what the base model can reliably express from its parameters alone (Guo et al., 2025b; Wang et al., 2026; Yim et al., 2026).

Despite this progress, existing skill-based agentic RL methods largely follow two monotonic paradigms. One paradigm treats skills as persistent augmentation and continuously expands the external skill bank to support exploration and decision-making (Xia et al., 2026). The other treats skills as temporary scaffolds and gradually removes them toward zero-skill inference, aiming to transfer their benefits into model parameters (Lu et al., 2026). While effective in their respective settings, both approaches implicitly assume that the active external skill set should either keep growing or eventually disappear. This assumption overlooks a more general question: *As the agent learns, how should its active external skill set evolve under limited parametric capacity and uneven marginal contributions across skills?*

This question is especially important because parametric storage in language models is finite and constrained by model size, training budget, and the trade-off between memorization and generalization (Allen-Zhu & Li, 2024; 2025a;b). As a result, not every useful capability should be forced into model parameters. External skills are particularly suitable for preserving narrow, low-frequency, or long-tail procedures that may be costly or unnecessary to encode parametrically (Zhang et al., 2026b). At the same time, keeping too many skills active is not free since large skill banks can introduce routing noise, and long injected contexts may reduce the reliability of skill use (Zheng et al., 2026; Liu et al., 2024b). Therefore, the central problem is not whether skills should be accumulated or eliminated, but how to determine the external boundary of a learning agent. A skill should be retained when it still provides marginal external value, retired when its contribution becomes negligible, and expanded when persistent failures reveal missing capability coverage.

To address this problem, we propose SLIM, a framework for dynamic **S**kill **LI**fecycle **M**anagement in agentic reinforcement learning (RL). SLIM dynamically updates the active external skill set as the policy evolves. Specifically, SLIM maintains a task-conditioned active skill set during RL, retrieves hierarchical skills from the current active pool, estimates the marginal external contribution (MEC) of each active skill through leave-one-skill-out validation, and couples these signals with RL-based policy optimization to retain, retire, or expand the active skill set over training. This creates a practical management mechanism between model parameters and external modular skills: reusable capabilities can be absorbed by the policy when external support becomes unnecessary, while narrow or long-tail capabilities can remain external when they continue to provide value. As shown in Figure 1, SLIM yields a non-monotonic external capability trajectory rather than forcing full accumulation or zero-skill inference adopted in SkillRL or Skill0.

We evaluate SLIM with Qwen3-4B on ALFWorld and SearchQA against standard GRPO (Guo et al., 2025a) and representative skill augmentation and internalization methods, including SkillRL (Xia et al., 2026) and Skill0 (Lu et al., 2026). SLIM outperforms the best baselines by an average of 7.1 percentage points. Additional experiments introduce Qwen3-8B as a new policy backbone and WebShop as a new benchmark (Appendix D.1). Across these settings, SLIM learns policy- and task-dependent endpoints, retaining compact skill sets on ALFWorld and WebShop while approaching zero external support on SearchQA.

Our contributions are threefold. (i) We formulate skill-based agentic RL as a dynamic skill lifecycle management problem, where the active external skill set is not assumed to monotonically grow or vanish, but is treated as an adaptive external capability boundary. (ii) We propose SLIM, which couples GRPO policy learning with MEC-guided retain, retire, and expand updates, adapting external support as the capabilities of the policy evolve. (iii) Experiments with two policy backbones across three benchmarks show that SLIM improves task performance while learning a policy- and task-dependent boundary between internalized capabilities and external skills.

2 Related Work

Large Language Model Agents. Large language model (LLM) agents turn autoregressive models into sequential decision makers that plan, act, and interact with external environments through tools, APIs, and embodied interfaces (Yao et al., 2023; Xie et al., 2024; Schick et al., 2023). Progress in tool use (Patil et al., 2024; Qin et al., 2024; Hao et al., 2023), web navigation (He et al., 2024;

OpenAI, 2025b; Google, 2025), computer use (OpenAI, 2025a; Anthropic, 2024), and long-horizon task completion (Zhao et al., 2026; Chen et al., 2025; Jin et al., 2025) shows that structured action spaces and external scaffolding are crucial for reliable agent behavior. External memory (Xu et al., 2025b; Chhikara et al., 2025) and skill support (Chen et al., 2026; Ni et al., 2026; Wang et al., 2026; Zheng et al., 2025) further improve robustness and compositionality. Our work follows this line but focuses on how the active external skill set should evolve during RL training.

Agentic Reinforcement Learning. Reinforcement learning has become a key paradigm for post-training LLM agents (Xu et al., 2025a; Lin et al., 2025), especially when interaction, exploration, and delayed credit assignment are required (Wei et al., 2025; Goldie et al., 2025; Zhao et al., 2025). Recent methods combine policy optimization with structured rewards, preference signals, or group-relative objectives to improve reasoning and action quality (Guo et al., 2025a; Shen et al., 2025; Schulman et al., 2017; Fujimoto et al., 2019; Yu et al., 2025). These advances provide a strong optimization backbone, but they do not determine how external skills should be retained, removed, or expanded during training. SLIM keeps the RL optimizer fixed and studies this external capability-management problem as the policy continues to learn.

Skill-Based Agents. Skill is a long-standing mechanism for organizing reusable agent behavior into modular procedures for task execution (Zhou et al., 2026; Zhang et al., 2026b). Recent LLM-agent work instantiates this idea through external skill banks (Wang et al., 2023; 2024b; Zheng et al., 2025; Wang et al., 2026; Xia et al., 2026), reusable prompt modules (Fu et al., 2024; Liu et al., 2024a; Li et al., 2025b; Ye et al., 2025), and distilled procedural guidance (Ni et al., 2026; Chen et al., 2026; Mi et al., 2026). Closely related methods either keep skills as persistent augmentation (Xia et al., 2026), eliminate them toward zero-skill inference (Lu et al., 2026), or continuously discover and refine reusable skills from rollout experience (Wu et al., 2026). SLIM is complementary to these directions, *i.e.*, it dynamically updates the active external skill set as the RL policy evolves and decides when skills should be retained, retired, or expanded under finite model capacity.

3 PRELIMINARIES

LLM Agent. We model an LLM agent as a policy π_θ that interacts with an environment over sequential decisions. Given a task instance $x \sim \mathcal{X}$, the agent produces a trajectory $\tau = (o_1, a_1, \ldots, o_T, a_T)$, where o_t and a_t are the observation and action at step t, and T is the horizon. The policy $\pi_\theta(a_t \mid h_t)$, parameterized by θ, conditions on the history $h_t = (x, o_1, a_1, \ldots, o_t)$. In text-only environments, both o_t and a_t are token sequences, and π_θ is a causal language model that autoregressively generates the next action from h_t.

Group Relative Policy Optimization. We use Group Relative Policy Optimization (GRPO) (Guo et al., 2025a) as the RL optimizer. For each task x, GRPO samples G trajectories $\{\tau^{(g)}\}_{g=1}^{G}$ from the behavior policy $\pi_{\theta_{\text{old}}}$ and assigns each a scalar reward $R(\tau^{(g)})$. Let $\mathbf{r} = \{R(\tau^{(1)}), \ldots, R(\tau^{(G)})\}$. The group-relative normalized advantage is $\hat{A}^{(g)} = \frac{R(\tau^{(g)}) - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$. Since rewards are outcome-level, the same $\hat{A}^{(g)}$ is used for all action-generation steps in $\tau^{(g)}$. Let $T^{(g)}$ be the number of action steps and $\rho_t^{(g)}(\theta) = \frac{\pi_\theta(a_t^{(g)} | h_t^{(g)})}{\pi_{\theta_{\text{old}}}(a_t^{(g)} | h_t^{(g)})}$ be the step-wise policy ratio. The GRPO objective is

$$J_{\text{GRPO}}(\theta) = \mathop{\mathbb{E}}_{\substack{x \sim \mathcal{X}, \\ \{\tau^{(g)}\}_{g=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot|x)}} \left[\frac{1}{G} \sum_{g=1}^{G} \frac{1}{T^{(g)}} \sum_{t=1}^{T^{(g)}} \left(\min \left(\rho_t^{(g)}(\theta) \hat{A}^{(g)}, \text{clip}\left(\rho_t^{(g)}(\theta), 1-\epsilon, 1+\epsilon \right) \hat{A}^{(g)} \right) - \beta \, \mathbb{D}_{\text{KL}}[\pi_\theta \,\|\, \pi_{\text{ref}}](h_t^{(g)}) \right) \right], \quad (1)$$

where ϵ and β are hyper-parameters and π_{ref} is the reference policy.

Skill Bank and Problem Setting. Following SkillRL (Xia et al., 2026), we assume a hierarchical external skill library with *general skills* and *task-specific skills*. Let $\mathcal{S}$ denote the global skill bank, with general-skill pool $\mathcal{S}^{\text{gen}}$ and task-specific pool $\mathcal{S}^k$ for task type k. At audit step t, the agent only accesses an active subset $\mathcal{A}_t \subseteq \mathcal{S}$ and acts under a skill-conditioned policy $\pi_\theta(a_t \mid h_t, s)$, where s denotes the selected external skill. Skills outside $\mathcal{A}_t$ are inactive and do not condition the policy. As the policy improves during RL, the usefulness of external skills changes. The problem is to determine which skills should remain active, which can be removed, and when additional skills are needed to support the evolving policy.

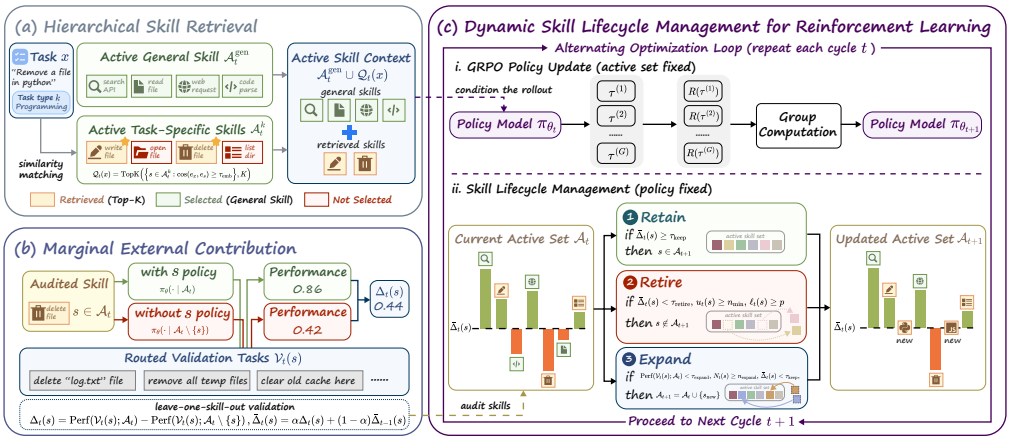


Figure 2: An overview of SLIM. SLIM first retrieves task-conditioned visible skills, then estimates skill-level marginal contribution via leave-one-skill-out validation, and finally updates the policy and skill lifecycle through GRPO-based retain–retire–expand operations.

4 METHOD: SLIM

An overview of SLIM is shown in Figure 2. SLIM addresses the skill management problem in Section 3 through three components. First, it restricts the active-set search to a task-conditioned set of visible skills (Section 4.1). Next, it estimates the local value of each audited skill through leave-one-skill-out validation (Section 4.2). Finally, it combines these signals with GRPO-based policy optimization, enabling the active skill set to be retained, retired, or expanded as training proceeds (Section 4.3). In this way, SLIM learns which capabilities should remain active and which should be removed from active external support.

4.1 HIERARCHICAL SKILL RETRIEVAL

The first component of SLIM restricts skill selection to task-conditioned candidates. Directly selecting from the full skill bank is a combinatorial problem, so SLIM uses the hierarchical setup in Section 3 to convert global skill selection into task-conditioned candidate selection.

Formally, let $\mathcal{A}_t^{\text{gen}} \subseteq \mathcal{S}^{\text{gen}}$ denote the currently active general-skill pool at audit step t, and let $\mathcal{A}_t^k \subseteq \mathcal{S}^k$ denote the active task-specific pool for task type k. For a task instance x of type k, SLIM selects the active general skills together with a retrieved task-specific subset from $\mathcal{A}_t^k$. Let e_x denote the embedding of the current task description and let e_s denote the embedding of skill s. The retrieved task-specific skill set is

$$\mathcal{Q}_t(x) = \text{TopK}\left(\left\{s \in \mathcal{A}_t^k : \cos(e_x, e_s) \geq \tau_{\text{emb}}\right\}, K\right), \tag{2}$$

where τ_{emb} is the retrieval threshold and K is the maximum number of task-specific skills loaded into the prompt. The final skill-conditioned policy for task x is thus the union of the active general skills and the retrieved task-specific set, *i.e.*, $\pi_\theta(a_t \mid h_t, \mathcal{A}_t^{\text{gen}} \cup \mathcal{Q}_t(x))$. Because retrieval is restricted to the current active set, lifecycle decisions directly affect the external capability exposed to later rollouts.

Intuitively, external skills must be relevant before they can be useful. At the same time, retrieval relevance alone does not tell us whether keeping a skill external is still worthwhile. Different active skills may be selected for the same type of tasks while contributing very different amounts of external value. This motivates an explicit estimate of the marginal external contribution of each active skill.

4.2 MARGINAL EXTERNAL CONTRIBUTION ESTIMATION

Given the routed skill set from Section 4.1, the next problem is to decide whether each active skill still deserves external support. Even after restricting the candidate set, enumerating skill combinations to estimate the marginal external contribution (MEC) of each active skill remains impractical. SLIM therefore uses leave-one-skill-out validation as a tractable local approximation.

For an audited skill $s \in \mathcal{A}_t$, let $\mathcal{V}_t(s)$ denote the subset of tasks from the development split whose rollouts use skill s under the current active set, *i.e.*, tasks x for which $s \in \mathcal{A}_t^{\text{gen}} \cup \mathcal{Q}_t(x)$. Let $\text{Perf}(\mathcal{V}; \mathcal{A})$ denote the validation performance on subset $\mathcal{V}$ when the active set is $\mathcal{A}$. The MEC of s at audit step t is defined by leave-one-skill-out validation:

$$\Delta_t(s) = \text{Perf}(\mathcal{V}_t(s); \mathcal{A}_t) - \text{Perf}(\mathcal{V}_t(s); \mathcal{A}_t \setminus \{s\}) . \tag{3}$$

To reduce audit noise, SLIM smooths current-round estimates with an exponential moving average, $\bar{\Delta}_t(s) = \alpha\Delta_t(s) + (1-\alpha)\bar{\Delta}_{t-1}(s)$. We use $\bar{\Delta}_t(s)$ rather than $\Delta_t(s)$ for lifecycle management. A positive value means the current policy still benefits from keeping that capability external, while a near-zero or negative value means the capability may have been absorbed, become redundant, or become harmful as an external aid. This is a local estimate conditioned on the current policy, active set, and routing behavior, not a global attribution over all possible skill subsets. It is reliable when tasks from the development split routed to s reflect the same local behavior seen during rollout; in that case, removing s on those tasks is a direct test of whether the policy still needs its external support. Lemma A.4 in Appendix A further explains this local surrogate.

4.3 Dynamic Skill Lifecycle Management for Reinforcement Learning

We now couple skill lifecycle updates with policy optimization through alternating updates. SLIM decomposes each audit cycle into GRPO policy updates with the active set fixed, followed by skill lifecycle management with the policy fixed.

In the GRPO stage, $\mathcal{A}_t$ is fixed, and the policy is updated using $J_{\text{GRPO}}(\theta; \mathcal{A}_t)$ under the current external support. This step may reduce the dependence of the policy on some external skills, but whether such dependence has actually disappeared is measured by MEC rather than assumed.

In the skill lifecycle management stage, θ_{t+1} is fixed. SLIM uses the smoothed MEC $\bar{\Delta}_t(s)$ and routed failures to retain, retire, or expand skills. Candidates are audited sequentially; after each retirement, routing and the validation baseline are refreshed before the next candidate is evaluated.

Retain keeps an audited skill s active when its smoothed MEC is clearly positive. Here τ_{keep} specifies the contribution level for retaining s as external support in later rollouts.

$$\text{if } \bar{\Delta}_t(s) \geq \tau_{\text{keep}}, \text{ then } s \in \mathcal{A}_{t+1}. \tag{4}$$

Retire removes an audited skill s when its marginal contribution becomes negligible and this signal remains stable after sufficient exposure. Here $u_t(s)$ is the cumulative exposure count and $\ell_t(s)$ is the low-contribution streak. These two conditions protect low-frequency skills from being removed before enough routed evidence is observed. The threshold τ_{retire} specifies the level below which the measured contribution is treated as negligible. Retiring s only means that it no longer provides enough marginal value under the current policy; it may have been internalized, become redundant, or become noisy or obsolete. When $\tau_{\text{retire}} \leq \bar{\Delta}_t(s) < \tau_{\text{keep}}$, SLIM makes no immediate lifecycle transition for s and keeps it active until later audits provide stronger evidence.

$$\text{if } \bar{\Delta}_t(s) < \tau_{\text{retire}}, \; u_t(s) \geq n_{\min}, \; \ell_t(s) \geq p, \text{ then } s \notin \mathcal{A}_{t+1}. \tag{5}$$

Expand adds a new skill s_{new} when the current active skill s persistently fails to cover its routed task region. Here $N_t(s)$ is the accumulated number of task failures routed to s. The threshold τ_{expand} identifies low-performing task regions whose failure traces guide the creation of new skills.

$$\text{if } \text{Perf}(\mathcal{V}_t(s); \mathcal{A}_t) < \tau_{\text{expand}}, \; N_t(s) \geq n_{\text{expand}}, \; \bar{\Delta}_t(s) < \tau_{\text{keep}}, \text{ then } \mathcal{A}_{t+1} = \mathcal{A}_t \cup \{s_{\text{new}}\}, \tag{6}$$

Theorem A.7 bounds the performance loss of sequential retirements under the audit accuracy and routing conditions in Appendix A, while Lemma A.5 characterizes when a currently useful skill is protected from retirement. Intuitively, if the policy still depends on a necessary skill, removing it hurts validation performance, $\bar{\Delta}_t(s)$ remains high, and retirement is blocked; if $\bar{\Delta}_t(s)$ stays near zero, active retention is unnecessary because the skill may have been internalized or become redundant. Additionally, SLIM subsumes prior methods as boundary cases. If retirement is disabled, i.e., $\mathcal{A}_{t+1} \supseteq \mathcal{A}_t$ for all t, it reduces to a SkillRL-like persistent augmentation regime. The active set then grows without removing skills whose contribution has diminished. If expansion is disabled and retirement is enforced until $\mathcal{A}_t = \varnothing$, it reduces to a Skill0-like zero-skill regime. This regime removes all external support, including skills that may still benefit the current policy.

5 IMPLEMENTATION

Algorithm. Algorithm 1 summarizes the practical training loop of SLIM. The implementation follows the three components in Section 4.1–4.3, *i.e.*, each GRPO step retrieves active skills, performs skill-conditioned rollouts, and updates the policy; every audit interval, SLIM estimates marginal external contribution and applies retain, retire, or expand operations. To keep auditing affordable, SLIM does not evaluate every active skill. Lifecycle audits are performed every $d = 10$ GRPO steps, and each audit considers at most $M = 4$ skills with the highest recent routed usage among skills that appeared in top-K retrieval.

Algorithm 1: Practical SLIM Training Loop

Require: Initial policy π_θ, skill bank $\mathcal{S}$, active set $\mathcal{A}_0$, train split, development split, retrieval cap K, audit interval d, task-specific audit budget M, expansion budget B
Ensure: Trained policy π_θ, final active set $\mathcal{A}_T$, retired skills, expanded skills, lifecycle logs

for GRPO step $r = 1, \ldots, T$ **do**
  Sample tasks, retrieve $\mathcal{Q}_t(x)$ by Eq. (2), and roll out $\pi_\theta(\cdot \mid h, \mathcal{A}_t^{\text{gen}} \cup \mathcal{Q}_t(x))$.
  Update θ with GRPO on the rollouts.
  if $r \bmod d = 0$ **then**
    Run validation with current routing; record routed skills, outcomes, and routed failures.
    Select audit candidates within budget, including top-M skills by recent routed usage.
    for each audited skill s **do**
      Compute leave-one-skill-out $\Delta_t(s)$ by Eq. (3).
      Update $\bar{\Delta}_t(s)$ and lifecycle statistics.
      Apply Eqs. (4)–(5); after retirement, refresh routing and the full-active-set validation baseline.
    end for
    For each qualifying failure bucket, create up to B task-specific skills by Eq. (6); update $\mathcal{A}_t$.
  end if
end for

Training and Inference Settings. For training, task-specific retrieval uses Qwen3-Embedding-0.6B (Zhang et al., 2025) with $K = 3$ and $\tau_{\text{emb}} = 0.45$. We optimize the policy with GRPO using outcome-level rewards. Specifically, each completed rollout receives the environment success reward, with invalid-action penalties applied during trajectory collection. In the main SLIM runs, we disable both policy-side KL loss and KL-in-reward regularization. Expansion uses o3 with an Anthropic-style skill-creator workflow (Anthropic, 2025) to create standalone task-specific `SKILL.md` artifacts from routed failure buckets, matching SkillRL's default creator choice. During final inference, the agent can run with skills by retrieving active skills before each rollout; the prompt contains the active general skills and the retrieved task-specific set $\mathcal{Q}_T(x)$, and no lifecycle update is performed. Prompt templates, lifecycle thresholds, full training settings, and inference details are provided in Appendix B.1.

6 EXPERIMENT

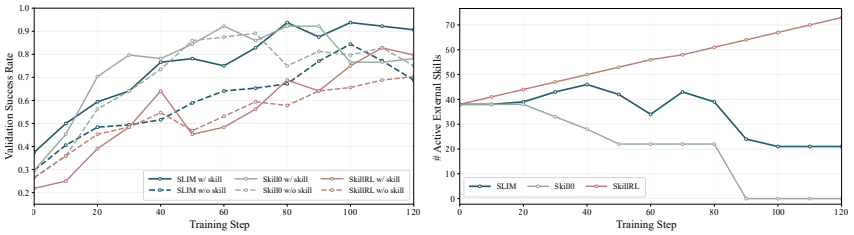
6.1 EVALUATION SETUP

Benchmarks Baselines. We conduct all main experiments with Qwen3-4B (Yang et al., 2025) on ALFWorld (Shridhar et al., 2021) and SearchQA (Jin et al., 2025). ALFWorld covers Pick, Look, Clean, Heat, Cool, and Pick2 household tasks, while SearchQA covers NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023) , HotpotQA (Yang et al., 2018), 2Wiki (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023). We compare against prompt-based, agent/memory-based, and RL-based baselines, including ReAct (Yao et al., 2023), Reflexion (Shinn et al., 2023), Mem0 (Chhikara et al., 2025), ExpeL (Zhao et al., 2024), GRPO (Shao et al., 2024), EvolveR (Wu et al., 2025), SkillRL (Xia et al., 2026), and Skill0 (Lu et al., 2026). Full baseline details are provided in Appendix B.2.

Evaluation. We report success rates under the same benchmark success criteria, i.e., objective completion on ALFWorld and correct final answers on SearchQA. Additionally, all methods share the same protocol. Policy updates use the train split, hyperparameters are selected on a smaller subset of the train split, skill management uses the development split, and final reporting uses the test split. Online-RL methods use the final checkpoint after a fixed training horizon, without cold-start SFT or warmup. Appendix D reports additional backbone and benchmark evaluations, independent training runs, skill-bank transfer, sensitivity, hyperparameters, expanded baselines, and audit overhead. Detailed splits and fairness controls are in Appendix C.

Table 1: Main results on ALFWorld and SearchQA. All entries report success rate. † denotes evaluation with retrieved external skills and applies throughout unless otherwise specified. Avg. denotes the overall success rate, also used below. **Best** and second-best are highlighted.

Method	ALFWorld							SearchQA							
	Pick	Look	Clean	Heat	Cool	Pick2	Avg.	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Avg.
Prompt-based methods															
Zero-Shot	82.9	0.0	31.2	23.1	21.7	30.0	41.4	28.5	49.6	32.7	23.5	27.7	5.3	35.2	32.3
Few-Shot	80.0	40.0	37.5	7.7	17.4	15.0	39.1	32.9	55.6	34.9	28.0	28.5	7.2	37.6	35.5
Zero-Shot†	94.3	40.0	81.2	38.5	17.4	55.0	63.3	26.6	48.2	31.8	22.7	27.9	5.3	33.6	31.5
Few-Shot†	94.3	60.0	84.4	38.5	34.8	30.0	64.1	32.3	54.6	34.6	28.1	27.2	7.4	**38.4**	34.8
Agent- or memory-based methods															
ReAct	**100.0**	33.3	25.0	53.8	23.8	26.7	50.5	33.2	54.5	36.5	28.8	30.8	7.7	33.6	36.4
Reflexion	93.9	30.0	52.0	20.0	47.8	68.2	59.4	22.3	45.7	25.8	22.6	28.3	4.9	31.2	29.1
Mem0	93.9	30.0	32.0	26.7	17.4	18.2	42.2	29.3	50.6	33.3	24.6	28.2	6.5	30.4	33.1
ExpeL	97.1	62.5	38.5	13.3	21.7	61.9	53.5	23.5	47.0	29.6	22.8	30.0	6.5	28.0	31.0
RL-based methods															
GRPO	85.4	**100.0**	49.8	64.6	53.1	54.2	67.2	35.9	57.8	36.5	30.8	30.1	9.2	30.4	37.5
GRPO†	84.6	62.5	48.9	68.8	61.0	80.8	68.8	36.4	58.2	37.0	31.2	30.4	9.6	31.2	37.9
EvolveR	67.6	37.5	49.3	15.6	36.0	36.5	39.8	36.0	57.5	36.8	30.6	30.0	9.5	29.6	37.4
SkillRL†	90.6	75.0	54.6	76.2	67.7	**87.5**	75.0	36.8	59.8	36.9	31.5	29.7	10.3	31.2	38.1
Skill0	93.6	87.5	59.8	66.7	57.6	78.4	74.2	37.9	59.5	38.6	32.7	**31.9**	10.3	32.8	39.3
SLIM	91.4	80.0	46.9	61.5	73.9	85.0	72.7	**38.6**	**62.2**	40.0	**37.2**	31.7	**12.8**	37.6	**41.0**
SLIM†	92.9	**100.0**	**91.4**	**78.3**	**88.5**	81.2	**87.5**	38.4	62.1	**40.4**	36.9	31.5	12.7	36.0	**41.0**



(a) Validation Success Rate on ALFWorld (b) The Number of Active External Skills on ALFWorld

Figure 3: Training dynamics on ALFWorld. Panel (a) compares with-skill and no-skill evaluation curves for SLIM, SkillRL and Skill0, where solid lines denote evaluation with external skills, while dashed lines denote without external skills. Panel (b) tracks the number of active skills over training.

6.2 Main Results

Overall Comparison. Table 1 reports the main comparison. On ALFWorld, SLIM† reaches 87.5, outperforming the strongest non-SLIM baseline, SkillRL†, by 12.5 points. It also substantially improves over GRPO, GRPO†, and Skill0, showing that the gain is not produced by ordinary RL, naive skill injection, persistent accumulation, or forced zero-skill inference alone. The same trained SLIM policy achieves 72.7 without skills and 87.5 with its final active skill set, indicating that ALFWorld contains long-horizon procedural behaviors where some capabilities remain better kept externally. SearchQA shows a different regime. SLIM and SLIM† both reach 41.0, exceeding Skill0 by 1.7 points. Their matching performance indicates that the gain is largely reflected in the trained policy, with little final external dependence. Together, these results show task-dependent endpoints: retained procedural support on ALFWorld and near-zero dependence on SearchQA. Across three independent training runs with each backbone, SLIM achieves the highest mean success rate on both benchmarks, with stable lifecycle trajectories on ALFWorld (Appendix D.4).

Detailed Analysis. The ALFWorld gains concentrate on procedural state-transformation tasks. SLIM† reaches 91.4 on Clean and 88.5 on Cool, far above SkillRL† and Skill0. This aligns with the lifecycle probe in Figure 5, *i.e.*, the clean-specific skill `cle_003` remains externally valuable, while several cooling skills are retired, suggesting that the gain comes from filtering procedural support rather than preserving every task-specific skill. Heat shows a smaller but consistent improvement, indicating that lifecycle management is most useful when procedures are compositional and unevenly covered. The advantage is not uniform across all task types. Simpler object-acquisition tasks such as Pick leave limited headroom for lifecycle management, while SLIM† reaches full success on Look by retaining skills that cover this task type. More importantly, GRPO† improves Pick2 and Cool but hurts or does not improve Look and Pick, showing that naive skill insertion is not uniformly beneficial. This supports our lifecycle control, where skills remain active only when their marginal effect is positive under the current policy. On SearchQA, the improvement is smaller but broadly distributed: SLIM or SLIM† is best or near-best on most subsets. Naive skill insertion can even hurt, as Zero-Shot†

and Few-Shot† fall below their no-skill counterparts. Thus, SearchQA reflects a lower-dependence regime where lifecycle-guided training improves the policy more than inference-time skill insertion, complementing ALFWorld where retained procedural skills remain valuable.

6.3 Training Dynamics

Figure 3 compares training dynamics on ALFWorld. SkillRL follows persistent accumulation where its active skill count grows from 38 to 73 throughout training. Although both its with-skill and no-skill validation curves improve, its final performance remains below SLIM. This shows that keeping more external skills does not necessarily yield better final performance, consistent with our motivation that large skill banks can introduce routing and context overhead. Skill0 exhibits the opposite pattern. Its active set decreases from 38 to 0. Its no-skill performance becomes strong in the later stage, indicating that the policy indeed learns from skill-conditioned training. However, once the active skill set reaches zero around epoch 90, validation drops from 92.2% to 76.6% in the following audit interval. This supports our claim that retirement is not equivalent to successful internalization since forced zero-skill inference may remove useful external support, especially for unstable, low-frequency or long-tail capabilities.

Table 2: Ablation study on ALFWorld. All variants are trained under the same settings and evaluation protocol as SLIM (with skill).

Method	ALFWorld Avg.
SLIM	**87.5**
w/o Retirement	73.4
w/o Expansion	78.9
Random Audit	68.8
Fixed Active Set Size	75.6

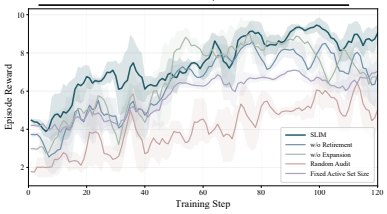


Figure 4: Training reward dynamics of SLIM and its ablation variants on ALFWorld. For readability, we apply a centered moving average with a window size of 5 training steps. The shaded region is the local variation within the window.

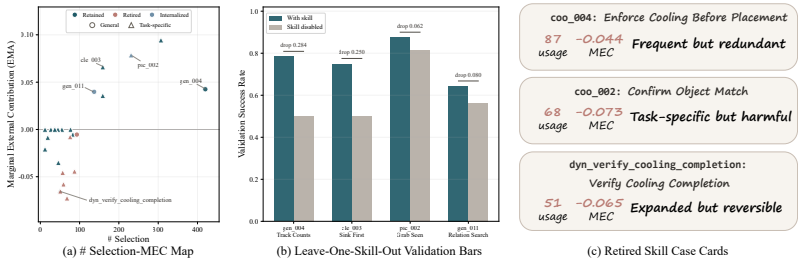
SLIM learns a non-monotonic trajectory. Its active set first expands from 38 to 46, fluctuates as expansion and retirement alternate, and finally stabilizes at a compact non-empty set of 21 skills. Meanwhile, no-skill performance rises from 29.7% to 84.4%, showing that the policy itself is learning, while with-skill performance peaks at 93.8% and remains 90.6% at the end. Thus, SLIM does not trade policy learning for external dependence. It improves the policy while preserving a set of external skills that still provide marginal contribution.

6.4 Ablation Study

Lifecycle Components. Table 2 isolates the lifecycle operations. Removing retirement drops ALFWorld success from 87.5 to 73.4, showing that expansion without deletion degenerates toward SkillRL-like accumulation. Without expansion, SLIM makes no creator calls and reaches 78.9, exceeding SkillRL† by 3.9 points. Expansion adds 8.6 points through new capability coverage, subsequent policy adaptation, and continued lifecycle management. Full SLIM preserves its advantage across creators, reaching 86.7 with frozen Qwen3-4B and 87.5 with o3 (Appendix D.5). Figure 4 shows the same pattern: local reward improvements can be transient, but the final endpoint remains worse without the full lifecycle.

Effectiveness of External Marginal Contribution. "Random Audit" keeps the same operation space but replaces contribution-aware decisions with stochastic ones: retain/delete is sampled with probabilities 0.8/0.2, and expansion is triggered independently with probability 0.1. It obtains only 68.8, the largest drop, and its reward curve stays below the other variants for most of training. Thus, the gain of SLIM does not come from random skill-set perturbation; lifecycle decisions must track whether a skill still provides marginal external value under the current policy.

Beyond Prompt-Budget Control. "Fixed Active Set Size" controls the active skill count at the initial size of 38, using LRU removal after expansion and expansion after retirement to keep the budget fixed. It reaches 75.6, still 11.9 points below SLIM. This rules out a pure prompt-budget explanation: the key is not only how many skills remain active, but which skills are retained, removed, and expanded.



(a) # Selection-MEC Map (b) Leave-One-Skill-Out Validation Bars (c) Retired Skill Case Cards

Figure 5: Case study of skill lifecycle on ALFWorld. Panel (a) plots selection count against marginal external contribution (MEC) for retained, retired, and internalized skills. Panel (b) reports leave-one-skill-out validation bars for representative retained skills. Panel (c) shows retired skill cases.

6.5 Case Study: Analysis of Skill Lifecycle Management

We use a lifecycle probe (Appendix B.1) to analyze SLIM operations. The probe logs each audited skill's selection count, marginal external contribution (MEC), with-skill performance, disabled-skill performance, and lifecycle outcome. Among frequently selected candidates, we label a skill as closer to *internalized* when its with-skill validation success rate is at least 60% and disabling it preserves at least 85% of that performance. This diagnostic label describes policy dependence, while lifecycle updates follow Section 4.3. While Figure 3 shows that the active set first expands, then contracts, and finally remains non-empty, Figure 5 explains how contribution-aware filtering yields this compact set of externally valuable skills.

Lifecycle Follows MEC. Panel (a) and (b) show that lifecycle decisions are governed by MEC rather than selection frequency alone. Broad and frequently selected skills such as `pic_002` and `gen_011` become close to internalized, as disabling them causes only 0.062 and 0.080 drops; however, high frequency is not sufficient for retirement, since `gen_004` is also broad and frequent but remains externally valuable with a much larger 0.284 drop. Conversely, low frequency does not imply low value: `cle_003` is less frequently selected but has high MEC, and disabling it causes a 0.250 drop, making it globally infrequent but locally indispensable. Panel (c) further shows that frequent, task-specific, or newly expanded skills can all be retired once their marginal external value becomes negative. Thus, SLIM retires a skill only when its external marginal value disappears under the current policy, while preserving long-tail skills that remain useful on specific routed subsets.

Empirical Support for Local Lifecycle Signals. The case study also connects the local theory view to observed behavior. The MEC probe shows that leave-one-skill-out drops align with retain and retire outcomes, while Figure 3 shows that these local decisions produce a non-monotonic active set rather than monotone accumulation or elimination. The ablations further support this mechanism: w/o Retirement falls to 73.4, Random Audit to 68.8, and Fixed Active Set Size to 75.6, indicating that performance depends on contribution-aware lifecycle decisions rather than unbounded skill growth, random mutation, or simple prompt-budget control.

7 Conclusion and Future Work

We present SLIM, a framework for dynamic skill lifecycle management in agentic RL. The key idea of SLIM is that external skills should neither be assumed to accumulate indefinitely nor be forced to vanish toward zero-skill inference. Instead, the active external skill set should be dynamically updated as the policy evolves, allowing the learned endpoint to avoid both persistent full accumulation and forced zero-skill inference. Across ALFWorld and SearchQA, SLIM improves over standard RL and prior skill-based methods while learning a qualitatively different endpoint. Training dynamics, ablations, and lifecycle case studies show that the active set evolves non-monotonically and that contribution-aware decisions are essential. These results suggest a division of labor between model parameters and external procedural memory: reusable behaviors can become less externally necessary, while narrow or locally consequential skills remain useful as external modules. Future work can extend this perspective to richer multimodal environments, finer-grained lifecycle units, and more scalable auditing methods.

AI USE STATEMENT

In this work, we used generative AI tools to support literature search, identify relevant work, generate external skill artifacts for agent training, assist with implementation, and refine the writing. The authors conceived the method, designed the experiments, and determined the interpretations and scientific claims. We reviewed the AI-assisted materials, including generated skills, code, and text, and take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Algorithm 1 summarizes the training procedure. Appendix B details the policy backbones, skill retrieval and lifecycle mechanisms, training hyperparameters, and baseline implementations. Appendix C specifies benchmark splits, checkpoint selection, and evaluation protocols, while Appendix E provides the prompts for environment interaction, skill insertion, baseline agents, and skill generation. Appendix F documents the skill banks. Appendix D reports additional analyses, including independent training runs, generalization, and sensitivity to skill initialization, creators, and lifecycle hyperparameters. The assumptions and proofs for the theoretical results are given in Appendix A.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE style optimization for learning from human feedback in LLMs. *arXiv preprint arXiv:2402.14740*, 2024.

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*, 2024.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.1, knowledge storage and extraction. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 1067–1077, Vienna, Austria, 2024.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.2, knowledge manipulation. In *Proceedings of the 13th International Conference on Learning Representations*, Singapore, Singapore, 2025a.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.3, knowledge capacity scaling laws. In *Proceedings of the 13th International Conference on Learning Representations*, Singapore, Singapore, 2025b.

Anthropic. Introducing computer use, a new claude 3.5 sonnet, and claude 3.5 haiku. `https://www.anthropic.com/news/3-5-models-and-computer-use`, 2024.

Anthropic. Agent skills. `https://docs.claude.com/en/docs/agents-and-tools/agent-skills`, 2025.

Le Chen, Erhu Feng, Yubin Xia, and Haibo Chen. Skvm: Revisiting language vm for skills across heterogenous llms and harnesses. *arXiv preprint arXiv:2604.03088*, 2026.

Xinyi Chen, Yilun Chen, Yanwei Fu, Ning Gao, Jiaya Jia, Weiyang Jin, Hao Li, Yao Mu, Jiangmiao Pang, Yu Qiao, Yang Tian, Bin Wang, Bolun Wang, Fangjing Wang, Hanqing Wang, Tai Wang, Ziqin Wang, Xueyuan Wei, Chao Wu, Shuai Yang, Jinhui Ye, Junqiu Yu, Jia Zeng, Jingjing Zhang, Jinyu Zhang, Shi Zhang, Feng Zheng, Bowen Zhou, and Yangkun Zhu. Internvla-m1: A spatially guided vision-language-action framework for generalist robot policy. *arXiv preprint arXiv:2510.13778*, 2025.

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.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin

Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

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. *arXiv preprint arXiv:2403.08978*, 2024.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2052–2062, Long Beach, CA, 2019.

Anna Goldie, Azalia Mirhoseini, Hao Zhou, Irene Cai, and Christopher D Manning. Synthetic data generation & multi-step rl for reasoning & tool use. *arXiv preprint arXiv:2504.04736*, 2025.

Google. Gemini Deep Research — your personal research assistant. `https://gemini.google/overview/deep-research/`, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025a.

Yijie Guo, Bingjie Tang, Iretiayo Akinola, Dieter Fox, Abhishek Gupta, and Yashraj Narang. SRSA: skill retrieval and adaptation for robotic assembly tasks. In *Proceedings of the 13th International Conference on Learning Representations*, Singapore, Singapore, 2025b. OpenReview.net.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. In *Advances in Neural Information Processing Systems 36*, New Orleans, LA, 2023.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 6864–6890, Bangkok, Thailand, 2024.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, Barcelona, Spain, 2020.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Proceedings of the 2nd Conference on Language Modeling*, Montreal, Canada, 2025.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pp. 1601–1611, Vancouver, Canada, 2017.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems 33*, pp. 9459–9474, 2020.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*, 2025a.

Zhigen Li, Jianxiang Peng, Yanmeng Wang, Yong Cao, Tianhao Shen, Minghui Zhang, Linxi Su, Shang Wu, Yihang Wu, YuQian Wang, Ye Wang, Wei Hu, Jianfeng Li, Shaojun Wang, Jing Xiao, and Deyi Xiong. Chatsop: An sop-guided mcts planning framework for controllable llm dialogue agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17637–17659, Vienna, Austria, 2025b.

Minhua Lin, Zongyu Wu, Zhichao Xu, Hui Liu, Xianfeng Tang, Qi He, Charu Aggarwal, Hui Liu, Xiang Zhang, and Suhang Wang. A comprehensive survey on reinforcement learning-based agentic search: Foundations, roles, optimizations, evaluations, and applications. *arXiv preprint arXiv:2510.16724*, 2025.

Anthony Zhe Liu, Jongwook Choi, Sungryull Sohn, Yao Fu, Jaekyeom Kim, Dong-Ki Kim, Xinhe Wang, Jaewon Yu, and Honglak Lee. Skillact: Using skill abstractions improves llm agents. In *ICML 2024 Workshop on LLMs and Cognition*, Vienna, Austria, 2024a.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for LLM agents. *arXiv preprint arXiv:2601.02553*, 2026.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024b.

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.

Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, Rongcheng Tu, Xiao Luo, Wei Ju, Zhiping Xiao, Yifan Wang, Meng Xiao, Chenwu Liu, Jingyang Yuan, Shichang Zhang, Yiqiao Jin, Fan Zhang, Xian Wu, Hanqing Zhao, Dacheng Tao, Philip S. Yu, and Ming Zhang. Large language model agent: A survey on methodology, applications and challenges. *arXiv preprint arXiv:2503.21460*, 2025.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pp. 9802–9822, Toronto, Canada, 2023.

Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. Skill-pro: Learning reusable skills from experience via non-parametric ppo for llm agents. *arXiv preprint arXiv:2602.01869*, 2026.

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

OpenAI. Computer-using agent. `https://openai.com/index/computer-using-agent/`, 2025a.

OpenAI. Deep research system card. `https://openai.com/index/deep-research-system-card`, 2025b.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems 37*, Vancouver, Canada, 2024.

Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Bäck. Multi-step reasoning with large language models, a survey. *ACM Computing Surveys*, 58(6): 160:1–160:35, 2026.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, Singapore, Singapore, 2023.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *Proceedings of the 12th International Conference on Learning Representations*, Vienna, Austria, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems 36*, New Orleans, LA, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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.

Junhao Shen, Haiteng Zhao, Yuzhe Gu, Songyang Gao, Kuikun Liu, Haian Huang, Jianfei Gao, Dahua Lin, Wenwei Zhang, and Kai Chen. Semi-off-policy reinforcement learning for vision-language slow-thinking reasoning. In *Advances in Neural Information Processing Systems 38*, San Diego, CA, 2025.

Zhuocheng Shen. Llm with tools: A survey. *arXiv preprint arXiv:2409.18807*, 2024.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36*, New Orleans, LA, 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 *Proceedings of the 9th International Conference on Learning Representations*, Virtual Conference, 2021.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

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.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024a.

Zhaoyang Wang, Qianhui Wu, Xuchao Zhang, Chaoyun Zhang, Wenlin Yao, Fazle Elahi Faisal, Baolin Peng, Si Qin, Suman Nath, Qingwei Lin, Chetan Bansal, Dongmei Zhang, Saravan Rajmohan, Jianfeng Gao, and Huaxiu Yao. Webxskill: Skill learning for autonomous web agents. *arXiv preprint arXiv:2604.13318*, 2026.

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

Quan Wei, Siliang Zeng, Chenliang Li, William Brown, Oana Frunza, Wei Deng, Anderson Schneider, Yuriy Nevmyvaka, Yang Katie Zhao, Alfredo Garcia, and Mingyi Hong. Reinforcing multi-turn reasoning in llm agents via turn-level reward design. *arXiv preprint arXiv:2505.11821*, 2025.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, and Botian Shi. Evolver: Self-evolving llm agents through an experience-driven lifecycle. *arXiv preprint arXiv:2510.16079*, 2025.

Xiyang Wu, Zongxia Li, Guangyao Shi, Alexander Duffy, Tyler Marques, Matthew Lyle Olson, Tianyi Zhou, and Dinesh Manocha. Co-evolving llm decision and skill bank agents for long-horizon tasks. *arXiv preprint arXiv:2604.20987*, 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.

Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, Leo Z. Liu, Yiheng Xu, Hongjin Su, Dongchan Shin, Caiming Xiong, and Tao Yu. Openagents: An open platform for language agents in the wild. In *Proceedings of the 1st Conference on Language Modeling*, Philadelphia, PA, 2024.

Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Chenyang Shao, Yuwei Yan, Qinglong Yang, Yiwen Song, Sijian Ren, Xinyuan Hu, Yu Li, Jie Feng, Chen Gao, and Yong Li. Towards large reasoning models: A survey of reinforced reasoning with large language models. *arXiv preprint arXiv:2501.09686*, 2025a.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025b.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang,

Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, 2018.

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 35*, pp. 20744–20757, New Orleans, LA, 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, Kigali, Rwanda, 2023.

Anbang Ye, Qianran Ma, Jia Chen, Muqi Li, Tong Li, Fujiao Liu, Siqi Mai, Meichen Lu, Haitao Bao, and Yang You. Sop-agent: Empower general purpose ai agent with domain-specific sops. *arXiv preprint arXiv:2501.09316*, 2025.

Tik Yu Yim, Wenting Tan, Sum Yee Chan, Tak-Wah Lam, and Siu Ming Yiu. Asda: Automated skill distillation and adaptation for financial reasoning. *arXiv preprint arXiv:2603.16112*, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026a.

Xing Zhang, Guanghui Wang, Yanwei Cui, Wei Qiu, Ziyuan Li, Bing Zhu, and Peiyang He. Experience compression spectrum: Unifying memory, skills, and rules in llm agents. *arXiv preprint arXiv:2604.15877*, 2026b.

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

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

Haiteng Zhao, Junhao Shen, Yiming Zhang, Songyang Gao, Kuikun Liu, Tianyou Ma, Fan Zheng, Dahua Lin, Wenwei Zhang, and Kai Chen. Achieving olympia-level geometry large language model agent via complexity boosting reinforcement learning. In *Proceedings of the 14th International Conference on Learning Representations*, Rio de Janeiro, Brazil, 2026.

Qingfei Zhao, Ruobing Wang, Dingling Xu, Daren Zha, and Limin Liu. R-search: Empowering llm reasoning with search via multi-reward reinforcement learning. *arXiv preprint arXiv:2506.04185*, 2025.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025.

YanZhao Zheng, ZhenTao Zhang, Chao Ma, YuanQiang Yu, JiHuai Zhu, Yong Wu, Tianze Xu, Baohua Dong, Hangcheng Zhu, Ruohui Huang, and Gang Yu. Skillrouter: Skill routing for llm agents at scale. *arXiv preprint arXiv:2603.22455*, 2026.

Chenyu Zhou, Huacan Chai, Wenteng Chen, Zihan Guo, Rong Shan, Yuanyi Song, Tianyi Xu, Yingxuan Yang, Aofan Yu, Weiming Zhang, Congming Zheng, Jiachen Zhu, Zeyu Zheng, Zhuosheng Zhang, Xingyu Lou, Changwang Zhang, Zhihui Fu, Jun Wang, Weiwen Liu, Jianghao Lin, and Weinan Zhang. Externalization in llm agents: A unified review of memory, skills, protocols and harness engineering. *arXiv preprint arXiv:2604.08224*, 2026.

The appendix is organized as below:
• Appendix A presents the local theoretical analysis that complements SLIM.
• Appendix B describes the detailed settings of SLIM and baselines.
• Appendix C describes the detailed experimental setup, including training configuration, retrieval settings, and lifecycle auditing details.
• Appendix D provides additional experiments on generalization, skill-bank transfer, training robustness, creator and hyperparameter sensitivity, and audit overhead.
• Appendix E introduces all prompts used in training, evaluation, and skill expansion.
• Appendix F summarizes the initial and expanded skill banks used by SLIM.
• Appendix G discusses the limitations of SLIM.

A Theoretical Analysis

This section formalizes the skill management problem in Section 3 and analyzes the MEC-based lifecycle rules. We connect the motivation for persistent external support to knowledge representation, derive an error bound for smoothed MEC, and bound the performance loss of sequential retirements. The lifecycle analysis uses expected success rates on a fixed task distribution for validation and follows the policy and active set at each audited decision.

A.1 Parametric Knowledge and External Support

The Physics of Language Models studies distinguish knowledge storage, extraction, and manipulation. Part 3.3 develops bit-complexity bounds and studies how knowledge storage depends on model size and training exposure in controlled factual-knowledge settings (Allen-Zhu & Li, 2025b). Parts 3.1 and 3.2 show that storing knowledge and reliably extracting or manipulating it are distinct requirements (Allen-Zhu & Li, 2024; 2025a). This distinction motivates evaluating an external skill through its contribution to the current policy's behavior. A simple counting argument makes the representation issue precise.

Proposition A.1 (Finite Representation)**.** *Consider a controlled knowledge-recovery problem with fixed background information. Let Z be uniformly distributed over N target knowledge states, and let a training procedure encode Z into a parameter state W taking values in a finite set $\mathcal{W}$. A decoder that receives only W and the fixed background produces $\widehat{Z}$, possibly using independent randomness. Then*

$$\Pr(\widehat{Z} = Z) \le \min\left\{1, \frac{|\mathcal{W}|}{N}\right\}. \tag{A1}$$

Consequently, recovery with error at most $\epsilon \in [0, 1)$ requires $|\mathcal{W}| \ge (1 - \epsilon)N$.

Proof. Write $p(w \mid z)$ for the encoder distribution and $d(z \mid w)$ for the decoder distribution. Since the decoder accesses the target state only through W,

$$\begin{aligned}\Pr(\widehat{Z} = Z) &= \frac{1}{N} \sum_{w \in \mathcal{W}} \sum_{z=1}^{N} p(w \mid z) d(z \mid w) \\ &\le \frac{1}{N} \sum_{w \in \mathcal{W}} \sum_{z=1}^{N} d(z \mid w) \le \frac{|\mathcal{W}|}{N}.\end{aligned} \tag{A2}$$

The probability is also at most one. Rearranging $1 - \epsilon \le |\mathcal{W}|/N$ gives the necessary condition. □

Finite-precision parameterizations have finitely many states, so this observation applies to their representation of a controlled knowledge family. External artifacts provide an additional input through which task-relevant information can be supplied. Together with the distinction between storage and usable knowledge, this motivates a behavioral criterion: external support remains useful when disabling it reduces task performance. The following analysis quantifies this criterion directly through MEC.

A.2 Task-Conditioned Contribution

Let $\mathcal{D}$ be a fixed distribution over tasks used for validation, such as the uniform empirical distribution on the development split. For a policy π_θ and active set $\mathcal{A}$, define

$$P_\theta(x, \mathcal{A}) := \mathbb{E}[\mathrm{Perf}(x; \pi_\theta, \mathcal{A})] \in [0, 1], \qquad F_{\mathcal{D}}(\theta, \mathcal{A}) := \mathbb{E}_{x\sim\mathcal{D}}[P_\theta(x, \mathcal{A})], \tag{A3}$$

where the first expectation includes rollout randomness. Thus the bounds below concern performance on the specified audit distribution; test split performance is evaluated by the experiments.

Fix an audit after the policy update to θ_{t+1}, and let $\mathcal{A}_t$ denote the active set immediately before the skill comparison. Fix the retrieval rule and its tie-breaking. For an audited skill $s \in \mathcal{A}_t$, define the pre-ablation routing event and its probability by

$$R_t(s) = \{x : s \in \mathcal{A}_t^{\mathrm{gen}} \cup \mathcal{Q}_t(x)\}, \qquad q_t(s) = \Pr_{x\sim\mathcal{D}}(R_t(s)) > 0. \tag{A4}$$

The routed expected MEC is

$$\Delta_t^{\mathrm{loc}}(s) := \mathbb{E}_{x\sim\mathcal{D}}\left[P_{\theta_{t+1}}(x, \mathcal{A}_t) - P_{\theta_{t+1}}(x, \mathcal{A}_t \setminus \{s\}) \mid R_t(s)\right]. \tag{A5}$$

The routing event is held fixed for the paired comparison, while the counterfactual rollout retrieves from the remaining active set. Therefore, the difference includes the effect of any replacement skill. It conditions on all other active skills and allows their contributions to interact.

Lemma A.2 (From Routed to Aggregate Contribution)**.** *If disabling s leaves expected performance unchanged outside its pre-ablation routing event,*

$$P_{\theta_{t+1}}(x, \mathcal{A}_t) = P_{\theta_{t+1}}(x, \mathcal{A}_t \setminus \{s\}) \quad \textit{for } \mathcal{D}\textit{-almost every } x \notin R_t(s), \tag{A6}$$

then

$$F_{\mathcal{D}}(\theta_{t+1}, \mathcal{A}_t) - F_{\mathcal{D}}(\theta_{t+1}, \mathcal{A}_t \setminus \{s\}) = q_t(s)\Delta_t^{\mathrm{loc}}(s). \tag{A7}$$

Proof. Split the expected performance difference over $R_t(s)$ and its complement. The first term is $q_t(s)\Delta_t^{\mathrm{loc}}(s)$ by definition, and the second term is zero by Eq. (A6). □

Equation (A6) holds when removing an unselected skill leaves the retrieved context and rollout protocol unchanged. Deterministic top-K retrieval over fixed skill scores has this property for skills outside the selected set. This condition makes explicit when the controller's local comparison also determines its aggregate effect on $\mathcal{D}$.

A.3 Audit Error and EMA Tracking

For a fixed skill, index its successive audits by $j = 1, \ldots, J$. Write $\widehat{\Delta}_j$ for the raw estimate in Eq. (3), μ_j for its current routed expected MEC, and $\overline{\Delta}_j$ for its EMA. The policy, active set, and routed task subset may change between these audits.

Assumption A.3 (Audit Accuracy and Contribution Drift)**.** For a finite audit horizon, choose error budgets $\delta_j \in (0, 1)$ before the comparisons, with total budget below one. Conditional on the history $\mathcal{H}_{j-1}$ that selects the policy, active set, and skill for comparison j, the raw estimate satisfies

$$\Pr\left(|\widehat{\Delta}_j - \mu_j| > \sigma_j \mid \mathcal{H}_{j-1}\right) \le \delta_j. \tag{A8}$$

The change in routed expected contribution satisfies $|\mu_j - \mu_{j-1}| \le d_j$, and the initial estimate satisfies $|\overline{\Delta}_0 - \mu_0| \le e_0$. The nonnegative bounds σ_j, d_j, e_0 describe estimation error, contribution drift, and initialization error, respectively.

A sufficient sampling condition for Eq. (A8) is that, conditional on the selected comparison, $\widehat{\Delta}_j$ averages n_j independent paired success differences $Y_{j,r} \in [-1, 1]$ from the routed task distribution. Fresh task and rollout draws, or a complete independently rolled-out pass over the corresponding empirical task set, give average conditional mean μ_j. Hoeffding's inequality (Hoeffding, 1963) then gives

$$\Pr\left(|\widehat{\Delta}_j - \mu_j| > z \mid \mathcal{H}_{j-1}\right) \le 2\exp(-n_j z^2/2), \qquad \sigma_j = \sqrt{\frac{2\log(2/\delta_j)}{n_j}}. \tag{A9}$$

Here n_j counts independent evaluation pairs for that comparison; the cumulative exposure $u_t(s)$ is a separate eligibility statistic. For reused rollouts, the required accuracy bound concerns the resulting joint sampling scheme. The conditional formulation in Assumption A.3 accounts for adaptive candidate selection.

Lemma A.4 (Tracking the Current MEC)**.** *Under Assumption A.3, let* $0 < \alpha \leq 1$ *and use the EMA update from Section 4.2. Define*

$$e_j = (1-\alpha)(e_{j-1} + d_j) + \alpha\sigma_j. \tag{A10}$$

With probability at least $1 - \sum_{j=1}^{J} \delta_j$*, all* J *estimates satisfy* $|\overline{\Delta}_j - \mu_j| \leq e_j$*. If* $\sigma_j \leq \sigma$ *and* $d_j \leq d$*, then*

$$e_j \leq (1-\alpha)^j e_0 + \left(1 - (1-\alpha)^j\right)\left(\sigma + \frac{1-\alpha}{\alpha} d\right). \tag{A11}$$

Proof. Taking expectations over the history in Eq. (A8) and applying a union bound gives an event of probability at least $1 - \sum_j \delta_j$ on which every raw error is at most σ_j. On this event,

$$\begin{aligned} \overline{\Delta}_j - \mu_j &= (1-\alpha)(\overline{\Delta}_{j-1} - \mu_{j-1}) + (1-\alpha)(\mu_{j-1} - \mu_j) + \alpha(\widehat{\Delta}_j - \mu_j), \\ |\overline{\Delta}_j - \mu_j| &\leq (1-\alpha)(e_{j-1} + d_j) + \alpha\sigma_j = e_j. \end{aligned} \tag{A12}$$

Induction establishes the simultaneous bound. Summing the geometric recursion with $\sigma_j \leq \sigma$ and $d_j \leq d$ yields Eq. (A11). □

The error bound separates sampling noise from the effect of earlier policy and skill-set states. Refreshing a raw MEC estimate after a retirement updates the comparison to the new active set; the drift term accounts for older information retained by the EMA. No independence across smoothed estimates is needed for the simultaneous bound. Minimum exposure and patience add eligibility requirements to the retirement rule, so the bounds below apply to the subset of decisions that satisfy those requirements.

A.4 Retention and Sequential Retirement

At a particular audited decision, abbreviate the error bound from Lemma A.4 as $\varepsilon_t(s)$, so that $|\bar{\Delta}_t(s) - \Delta_t^{\mathrm{loc}}(s)| \leq \varepsilon_t(s)$ on its accuracy event.

Lemma A.5 (Protection of Useful Skills)**.** *On the audit accuracy event, any currently audited active skill satisfying*

$$\Delta_t^{\mathrm{loc}}(s) \geq \tau_{\mathrm{retire}} + \varepsilon_t(s) \tag{A13}$$

cannot be retired by Eq. (5)*. If Eq.* (4) *is triggered, its routed expected contribution is at least* $\tau_{\mathrm{keep}} - \varepsilon_t(s)$*.*

Proof. The first condition implies $\bar{\Delta}_t(s) \geq \Delta_t^{\mathrm{loc}}(s) - \varepsilon_t(s) \geq \tau_{\mathrm{retire}}$, which prevents the strict retirement inequality. For the retain rule, $\bar{\Delta}_t(s) \geq \tau_{\mathrm{keep}}$ implies $\Delta_t^{\mathrm{loc}}(s) \geq \bar{\Delta}_t(s) - \varepsilon_t(s) \geq \tau_{\mathrm{keep}} - \varepsilon_t(s)$. □

Lemma A.6 (Single-Retirement Loss)**.** *On the audit accuracy event, if Eq.* (5) *removes* s*, then*

$$\Delta_t^{\mathrm{loc}}(s) < \tau_{\mathrm{retire}} + \varepsilon_t(s). \tag{A14}$$

If Eq. (A6) *also holds, the aggregate performance change satisfies*

$$F_{\mathcal{D}}(\theta_{t+1}, \mathcal{A}_t \setminus \{s\}) - F_{\mathcal{D}}(\theta_{t+1}, \mathcal{A}_t) \geq -q_t(s)\big(\tau_{\mathrm{retire}} + \varepsilon_t(s)\big). \tag{A15}$$

Proof. Retirement requires $\bar{\Delta}_t(s) < \tau_{\mathrm{retire}}$. Therefore $\Delta_t^{\mathrm{loc}}(s) \leq \bar{\Delta}_t(s) + \varepsilon_t(s) < \tau_{\mathrm{retire}} + \varepsilon_t(s)$. Lemma A.2 expresses the aggregate change as $-q_t(s)\Delta_t^{\mathrm{loc}}(s)$, giving the bound. □

Theorem A.7 (Sequential Retirement)**.** *Fix the policy* θ *and audit distribution* $\mathcal{D}$ *during one lifecycle audit. Suppose the committed retirements are* $s_1, \ldots, s_m$*, with*

$$\mathcal{A}^{(i)} = \mathcal{A}^{(i-1)} \setminus \{s_i\}, \qquad i = 1, \ldots, m. \tag{A16}$$

For every removal, compute the routing probability q_i and the current MEC relative to $\mathcal{A}^{(i-1)}$, and let ε_i bound the corresponding smoothed estimation error. If Eq. (A6) *holds for each removal, then, on the simultaneous accuracy event,*

$$F_{\mathcal{D}}(\theta, \mathcal{A}^{(m)}) \geq F_{\mathcal{D}}(\theta, \mathcal{A}^{(0)}) - \sum_{i=1}^{m} q_i \big(\tau_{\text{retire}} + \varepsilon_i\big). \tag{A17}$$

The event has probability at least $1 - \sum_{a \in \mathcal{H}} \delta_a$ when Assumption A.3 holds over a finite audit horizon $\mathcal{H}$ covering the raw estimates used in these skills' EMA histories, with valid initialization and drift bounds.

Proof. Apply Lemma A.6 to each actual removal from its immediately preceding active set:

$$F_{\mathcal{D}}(\theta, \mathcal{A}^{(i)}) - F_{\mathcal{D}}(\theta, \mathcal{A}^{(i-1)}) \geq -q_i \big(\tau_{\text{retire}} + \varepsilon_i\big). \tag{A18}$$

Summing over i telescopes the left-hand side and proves Eq. (A17). Taking a union bound over the raw accuracy events in $\mathcal{H}$ and applying the EMA recursion gives the stated probability. The argument holds for the realized adaptive sequence, including its stopping point, on this simultaneous event. □

Each term in Eq. (A17) uses the active set produced by all earlier removals, so the result allows interactions among skills. For example, suppose two skills provide substitutable support. After the first is removed, if the second has MEC at least $\tau_{\text{retire}} + \varepsilon_i$ under the updated active set, Lemma A.5 prevents its subsequent retirement. Reusing both skills' pre-removal MEC values would not establish this conclusion. The cumulative bound pertains to the chosen audit order and the evaluated task distribution.

A.5 Policy Learning and Skill Expansion

GRPO changes the policy, while expansion changes the support available for future rollouts. To relate these stages to the retirement analysis, let $\mathcal{A}_t^-$ be the set after retirements and $\mathcal{A}_{t+1}$ the set after all accepted skill creations in the cycle. Define the signed performance changes

$$\begin{aligned} g_t^{\text{policy}} &= F_{\mathcal{D}}(\theta_{t+1}, \mathcal{A}_t) - F_{\mathcal{D}}(\theta_t, \mathcal{A}_t), \\ g_t^{\text{expand}} &= F_{\mathcal{D}}(\theta_{t+1}, \mathcal{A}_{t+1}) - F_{\mathcal{D}}(\theta_{t+1}, \mathcal{A}_t^-). \end{aligned} \tag{A19}$$

Here g_t^{expand} is the joint effect of the additions, including changes in retrieval. Adding and subtracting the intermediate performances and applying Theorem A.7 yields

$$F_{\mathcal{D}}(\theta_{t+1}, \mathcal{A}_{t+1}) - F_{\mathcal{D}}(\theta_t, \mathcal{A}_t) \geq g_t^{\text{policy}} + g_t^{\text{expand}} - \sum_{i=1}^{m} q_i \big(\tau_{\text{retire}} + \varepsilon_i\big). \tag{A20}$$

Thus, a cycle improves audit-distribution performance whenever the policy and expansion gains exceed the retirement loss bound. Equation (6) uses persistent routed failures to propose new skills; their contribution is assessed by subsequent audits. The quantities g_t^{policy} and g_t^{expand} describe performance changes in the analysis, while the implemented decisions use the GRPO objective, MEC, and failure statistics.

The resulting interpretation is behavioral: learning changes which external skills the policy benefits from, and lifecycle auditing tracks that changing dependence. Lemma A.5 preserves support whose current contribution exceeds the estimation margin; Theorem A.7 controls the loss of removing low-contribution support. These properties apply whether a low contribution arises from policy learning, substitutable skills, or ineffective guidance. The internalization probe in Appendix B.1 separately examines the policy's performance with and without particular skills.

B Implementation Details

B.1 SLIM Setup

Backbone and Data Protocol. All main SLIM experiments use Qwen3-4B (Yang et al., 2025) as the policy model. We use the train split for GRPO updates, the development split for lifecycle

auditing and training-time monitoring, and the test split only for final reporting. For ALFWorld, we use 16 tasks from the train split per update, 32 tasks from the development split per validation pass, a maximum prompt length of 4096, a maximum response length of 512, right truncation, and no overlength prompt filtering. For SearchQA, we use 64 tasks from the train split per update, 512 tasks from the development split per validation pass, a maximum prompt length of 5000, a maximum response length of 700, left truncation, and overlength prompt filtering. ALFWorld runs for 120 GRPO steps and SearchQA runs for 180 GRPO steps both with validation every 10 steps. Validation and final evaluation use sampled generation with temperature 0.4.

Training. SLIM uses the GRPO objective Eq. (1) and does not use a separate warmup or cold-start stage. ALFWorld uses $n = 8$ rollouts per prompt, a maximum of 50 environment steps, and 50 turns of interaction history. SearchQA uses $n = 4$ rollouts per prompt, a maximum of 4 environment steps, 4 turns of history, and the shared search backend used by all SearchQA methods. The optimizer uses learning rate 10^{-6} in both benchmarks. The PPO mini-batch and per-device micro-batch sizes are 32 and 2 for ALFWorld, and 512 and 8 for SearchQA. Rewards are outcome-level environment rewards: completed successful trajectories receive the benchmark success reward and failed trajectories receive zero reward, while invalid-action penalties are applied during trajectory collection. The invalid-action penalty coefficient is 0.1 for ALFWorld and 0.01 for SearchQA. For the main SLIM runs, both actor-side KL loss and KL-in-reward regularization are disabled.

Retrieval. SLIM uses the hierarchical SkillRL-style skill bank described in Section 3. Task-specific skills are retrieved only from the active pool of the detected task type. The retrieval query is the current task description, while each skill key is a routing text concatenating the skill title, description or principle, `when_to_apply` field, body, tags, and task type. We embed both query and keys with Qwen3-Embedding-0.6B (Zhang et al., 2025) and rank candidate skills by cosine similarity. We set the task-specific retrieval cap to $K = 3$ and the embedding threshold to $\tau_{\text{emb}} = 0.45$, so retrieval may insert fewer than three task-specific skills when no active skill is sufficiently relevant.

Skill Lifecycle. Lifecycle auditing runs periodically after GRPO validation, recording routed skills, validation outcomes, and routed failures under the current active set. With the policy fixed, candidates are audited sequentially. ALFWorld audits at most four recently routed task-specific skills from the top-K retrieved set, while SearchQA audits at most 12 active skills, supported by its larger validation batches and shorter episodes. Missing candidates are skipped, keeping each audit within its budget. For each candidate s, we record the performance of the current active set on its routed subset of the development split, temporarily disable s, and rerun the same tasks to compute the leave-one-skill-out marginal external contribution $\Delta_t(s)$. We then update the smoothed estimate $\bar{\Delta}_t(s)$ with EMA coefficient 0.9 and apply the retain and retire criteria. Retain decisions use $\tau_{\text{keep}} = 0.03$ for ALFWorld and $\tau_{\text{keep}} = 0.05$ for SearchQA. Retirement uses $\tau_{\text{retire}} = 0.001$ and patience $p = 3$ in both benchmarks, together with minimum exposure thresholds of $n_{\min} = 30$ for ALFWorld and $n_{\min} = 20$ for SearchQA. Thus, a skill is retired after sufficient routed exposure and persistently negligible contribution. Each triggered retirement is committed immediately, followed by refreshed skill routing and a recomputed full-active-set validation baseline for the next candidate. Subsequent MEC estimates reflect the updated active set. For two substitutable skills that jointly provide necessary coverage, retiring one removes their redundancy and allows the next audit to reveal the value of the remaining skill.

Skill Expansion. Expansion operates during training audits and adds task-specific skills from validation failure buckets, each grouping failures routed to the same active skill and task type. We select the expansion threshold together with the retention and retirement thresholds using performance on a smaller subset of the train split and keep them fixed across independent training runs of each setting. Both benchmarks use $\tau_{\text{expand}} = 0.40$. For each qualifying audited failure bucket, ALFWorld uses $n_{\text{expand}} = 20$ and creates at most $B = 2$ new skills, while SearchQA uses $n_{\text{expand}} = 15$ and creates at most $B = 3$. The sensitivity study in Appendix D.6 varies these three lifecycle thresholds one at a time on ALFWorld, with minimum exposure, patience, and the other training settings held fixed. For each selected failure bucket, an OpenAI-compatible `o3` skill-creator backbone generates standalone `SKILL.md` artifacts using an Anthropic-style skill-creator workflow (Anthropic, 2025). The expansion prompt in Figure A12 includes representative failed tasks, summarized failure traces, and the insufficient active skills. We deduplicate against existing skills by title, trigger description, and embedding similarity of the routing text, and reject generated skills that are generic, duplicate an existing skill, are too short to define a useful workflow, or attempt to create new general or

foundational skills. Accepted skills inherit the bucket's task type and enter the corresponding task-specific pool.

Audit Cost. Leave-one-skill-out auditing adds validation-time overhead, but it is bounded by the audit schedule and candidate budget. Audits run every 10 GRPO steps and use limited subsets of the development split. In ALFWorld, each audit evaluates at most one general-skill group and $M = 4$ task-specific skills; in SearchQA, the larger validation batch and shorter episodes allow at most 12 audited active skills. Each committed retirement additionally refreshes routing and the validation baseline for the remaining candidates. These refreshes are bounded by the number of audited candidates, keeping the validation cost linear in the audit budget. In our runs, ALFWorld completes in about 20 hours and SearchQA completes in about 25 hours, which is the same order as Skill0 and SkillRL under the shared training stack. Table A10 in Appendix D.8 provides the detailed overhead comparison.

Inference. For skill-conditioned final inference, SLIM uses the final active set after training. Before each rollout, the agent inserts active general skills and retrieves task-specific skills using the same task-type-scoped embedding retrieval as in training. Final evaluation does not perform retain, retire, or expand operations. The no-skill evaluation uses the same trained policy with both general and task-specific skill insertion disabled at inference. The environment prompts are shown in Figures A3 and A5, and the skill insertion format is shown in Figure A6.

Internalization Probe. For the ALFWorld lifecycle case study in Figure 5, we run a diagnostic probe every 20 training steps. Each probe selects up to five retained skills with at least 10 validation selections and utility EMA of at least 0.003. For each candidate s, $\mathrm{Perf}_{\mathrm{with}}$ is the success rate on its routed subset of the development split $\mathcal{V}_t(s)$ under the current policy and active set $\mathcal{A}_t$. We then rerun the same tasks with the policy fixed and only s disabled to obtain $\mathrm{Perf}_{\mathrm{without}}$. We label a skill as closer to internalized when $\mathrm{Perf}_{\mathrm{with}} \geq 0.60$ and $\mathrm{Perf}_{\mathrm{without}} \,/\, \mathrm{Perf}_{\mathrm{with}} \geq 0.85$. The label identifies skills for which the policy preserves at least 85% of its skill-conditioned performance after the skill is disabled. Probe outcomes are recorded for the case study, while training-time retain, retire, and expand decisions follow the lifecycle rules above.

Compute Resources. From logged token counts and step timings, the ALFWorld full SLIM run is on the order of 10^{19} floating-point operations. SearchQA uses shorter episodes but a larger validation batch and longer training horizon, giving an overall compute budget on the order of 10^{19}–10^{20} floating-point operations. These estimates include rollout generation, log-probability evaluation, policy updates, validation, and lifecycle audit reruns, but exclude one-time data preprocessing.

B.2 Baselines Setup

We use the original implementations of ReAct, Reflexion, ExpeL, Mem0, EvolveR, Skill0, and SkillRL with the benchmark configurations specified below.

Zero-Shot and Few-Shot Prompting. The prompt-only baselines evaluate the backbone model without RL updates. Zero-shot prompting directly uses the benchmark interaction prompt in Figure A2 or Figure A4. Few-shot prompting prepends solved examples following Figure A7. The with-skill variants use the same prompt protocol but additionally insert retrieved external skills using Figure A6. Prompt baselines are evaluated with the same environment parser, success metric, and test split as the RL methods, using batch size 16 for ALFWorld and 32 for SearchQA.

ReAct. ReAct interleaves reasoning and actions to solve tasks. It uses the instruction in Figure A8, preserves the same action tags and parser as the main environment, and does not use persistent memory.

Reflexion. Reflexion stores short reflections from failed trajectories and retrieves relevant reflections before later actions. The acting and reflection-generation prompts are shown in Figure A9. We retrieve at most three reflections and keep a bounded memory of 200 items. Generation uses deterministic service decoding with maximum 768 tokens, temperature 0, and top-$p = 1.0$.

ExpeL. ExpeL learns reusable lessons from past experience. After each episode, the method distills a compact reusable lesson from the trajectory and retrieves at most three relevant lessons for future decisions, with the same 200-item memory cap and deterministic service decoding as Reflexion.

Figure A10 gives the acting and lesson-distillation prompts. Evaluation uses the same action parser, success metric, and protocol for the train split, development split, and test split as the other baselines.

Mem0. Mem0 maintains long-term memory across tasks. It extracts up to three short atomic memories after each episode, stores them in the same bounded 200-item memory bank, and retrieves at most three relevant memories before action generation, following Figure A11.

GRPO and GRPO with Skills. The GRPO baseline uses the same RL optimizer, environment stack, and reward definition as SLIM, but does not receive external skill or memory context and does not use a warmup stage. ALFWorld uses train batch size 16, validation batch size 32, rollout count 8, maximum episode length 50, learning rate 10^{-6}, PPO mini-batch size 32, micro-batch size 2, 120 training steps, and validation every 10 steps. SearchQA uses train batch size 64, validation batch size 512, rollout count 4, maximum episode length 4, learning rate 10^{-6}, PPO mini-batch size 512, micro-batch size 8, 180 training steps, and validation every 10 steps. Both settings disable actor-side KL loss and KL-in-reward regularization, and use invalid-action penalty coefficients 0.1 for ALFWorld and 0.01 for SearchQA. GRPO with skills keeps the same optimizer and rollout settings but inserts retrieved external skills using the skill-conditioned prompt format in Figure A6; it does not perform retain, retire, or expand operations.

EvolveR. EvolveR is evaluated as an experience-lifecycle retrieval baseline. It distills past trajectories into reusable principles, retrieves relevant experience before rollout, and updates the experience store during training. We evaluate EvolveR on ALFWorld and SearchQA using the shared success-rate metric and protocol for the train split, development split, and test split. Where experience distillation or update requires a service model, we use the same OpenAI-compatible service family as the other agent and memory baselines. EvolveR retrieves the top-3 experience principles before each rollout. It does not estimate leave-one-skill-out marginal external contribution and does not maintain retain/retire/expand lifecycle states over skill artifacts.

Skill0. Skill0 follows its helpfulness-driven dynamic curriculum without a separate warmup stage in our comparison. Every d steps, Skill0 evaluates each skill file on its matched sub-task from the development split, computes its on-policy helpfulness as the performance difference between evaluation with and without that skill, filters out skills with non-positive helpfulness, ranks the remainder by helpfulness, and selects the top files under the current stage budget. In this Filter–Rank–Select procedure, helpfulness ranks skills within a predefined shrinking curriculum toward zero-skill inference. SLIM instead estimates smoothed MEC on each skill's currently routed subset of the development split and uses contribution, exposure, and failure signals to retain, retire, or expand a non-monotonic active set. ALFWorld uses the schedule $[6, 3, 0]$ with train batch size 16, validation batch size 32, rollout count 8, maximum episode length 50, learning rate 10^{-6}, PPO mini-batch size 32, micro-batch size 2, 120 training steps, and validation every 10 steps. SearchQA uses the schedule $[5, 3, 0]$ with train batch size 64, validation batch size 512, rollout count 4, maximum episode length 4, learning rate 10^{-6}, PPO mini-batch size 512, micro-batch size 8, and 180 training steps. Both settings disable actor-side KL loss and KL-in-reward regularization, and use the same invalid-action penalty coefficients as SLIM. Skill0 runs in the text-native ALFWorld and SearchQA environments with OCR and text rendering disabled.

SkillRL. SkillRL uses persistent skill-augmented RL and runs without a separate warmup stage in our comparison. The policy retrieves external skills during training and inference with the same SkillRL-style retrieval path used to initialize our skill bank, using the skill-conditioned prompt format in Figure A6. ALFWorld uses top-6 skill retrieval, dynamic skill-bank update with threshold 0.4, at most three new skills per update, train batch size 16, validation batch size 32, rollout count 8, maximum episode length 50, learning rate 10^{-6}, PPO mini-batch size 32, micro-batch size 2, 120 training steps, and validation every 10 steps. Its skill author/updater uses the OpenAI-compatible `o3` backbone that analyzes failed trajectories and emits new JSON skill records, matching the SkillRL-style skill-bank format rather than producing `SKILL.md` artifacts. SearchQA uses the same top-6 skill retrieval and dynamic skill-bank update protocol, with train batch size 64, validation batch size 512, rollout count 4, maximum episode length 4, learning rate 10^{-6}, PPO mini-batch size 512, micro-batch size 8, and 180 training steps. Unlike GRPO, SkillRL retains a reference-policy KL loss, using KL coefficients 0.01 for ALFWorld and 0.001 for SearchQA, while KL-in-reward remains disabled. Validation uses the dedicated development split, and the test split is reserved for final reporting.

B.3 Additional Backbone and Benchmark Setup

The cross-backbone experiments train SLIM, SkillRL, and Skill0 independently with Qwen3-4B and Qwen3-8B on each environment. Each run starts from the corresponding base policy checkpoint and learns its own policy and skill-set trajectory. ALFWorld and SearchQA use their respective benchmark configurations described above and in Appendix B.2.

For WebShop, training runs for 150 GRPO steps with a maximum of 15 environment actions per episode. The remaining RL settings follow the ALFWorld configuration of each method. SLIM starts with 57 active skills and applies task-conditioned retrieval and lifecycle auditing during training. The train split, development split, and test split contain separate product-shopping goals. Final evaluation reports binary success rate on goals in the test split using the final policy and skill set.

B.4 Additional Baselines Setup

GPT-4o and Gemini-2.5-Pro on ALFWorld. For the closed-source ALFWorld comparison, we report the GPT-4o and Gemini-2.5-Pro results from SkillRL (Xia et al., 2026). These results are used only in the expanded ALFWorld table in Appendix D.7 to contextualize the scale of agent performance under strong proprietary models. They are not used for hyperparameter selection, lifecycle auditing, or any training-time comparison.

SimpleMem on ALFWorld. SimpleMem is evaluated as a long-term memory baseline following its semantic memory compression and retrieval design (Liu et al., 2026). Each completed ALFWorld episode is converted into a compact textual memory record containing the task description, key observations, actions, outcome, and a short lesson. Memories are indexed with Qwen3-Embedding-0.6B to match the embedding backbone used elsewhere in our experiments. Following the original SimpleMem configuration style, we enable planning and reflection during retrieval, use semantic retrieval as the primary route, and allow keyword and structured filters as auxiliary routes. The memory retriever considers up to 25 semantic candidates, 5 keyword candidates, and 5 structured candidates, then inserts at most three compact memories into the action prompt to keep the prompt budget comparable to the other memory baselines. The policy model is Qwen3-4B, and the environment parser, success metric, train split, development split, test split, and final evaluation protocol are the same as the other baselines.

RLOO on ALFWorld. RLOO is evaluated as an RL optimizer baseline using the same ALFWorld environment stack as GRPO. The only optimizer-level change is the advantage estimator: for each prompt, we sample $K = 8$ rollouts and compute the leave-one-out advantage $\hat{A}_i = R_i - \frac{1}{K-1}\sum_{j \neq i} R_j$, following the REINFORCE leave-one-out formulation (Ahmadian et al., 2024). All other settings are matched to the ALFWorld GRPO baseline, including train batch size 16, validation batch size 32, maximum episode length 50, learning rate 10^{-6}, 120 training steps, validation every 10 steps, and no cold-start SFT. RLOO does not receive external skills, memory context, or lifecycle updates.

MemRL on ALFWorld. MemRL is evaluated as a runtime memory-reinforcement baseline that updates episodic memory without updating model parameters (Zhang et al., 2026a). We preserve the original method structure by using proceduralization for memory construction, query-based retrieval, and adjustment-based memory updates. To keep the comparison fair, the policy model is Qwen3-4B and the embedding model is Qwen3-Embedding-0.6B, while the ALFWorld parser, few-shot examples, test split, and success metric are shared with the rest of our evaluation. The memory retriever uses $k_{\text{retrieve}} = 5$, at most 8 extracted keywords, add-similarity threshold 0.90, novelty threshold 0.85, unknown-detection threshold 0.62, and a value-aware candidate set of size 3. The value update follows the original single-step setting with learning rate $\alpha = 0.3$, discount $\gamma = 0$, success reward 1.0, failure reward -1.0, and equal weights 0.5/0.5 for similarity and memory value in the combined retrieval score.

RAG on SearchQA. RAG is evaluated as a one-shot retrieval-augmented generation baseline on SearchQA (Lewis et al., 2020). Following the Search-R1 setting (Jin et al., 2025), each question is used as the retrieval query, the shared search corpus is queried before generation, and the top three retrieved passages are inserted into the prompt. The model then generates a final answer without iterative search calls, RL updates, skill retrieval, or lifecycle control. This baseline uses the same Qwen3-4B backbone, answer-matching rule, and test split as the other SearchQA methods.

Search-o1 on SearchQA. Search-o1 is evaluated as an inference-time agentic search baseline (Li et al., 2025a). The model is prompted to reason, issue search queries when needed, read retrieved evidence, and continue reasoning before producing the final answer. We use the same search interface, top-3 retrieved passages per search call, maximum action budget of 4 search rounds, and answer evaluator as the SearchQA environment. Search-o1 does not update model parameters and does not use external skills; it differs from RAG mainly by allowing iterative search and reasoning during inference.

Search-R1 on SearchQA. Search-R1 is evaluated as an RL-with-search baseline following the Search-R1 training protocol (Jin et al., 2025). During rollout, the policy alternates between model-generated reasoning tokens and search calls, and retrieved tokens are masked from the policy-gradient loss so that optimization is applied only to model-generated tokens. We use the shared SearchQA search interface with top-3 retrieved passages, maximum action budget 4, outcome exact-match reward, learning rate 10^{-6}, and the same separation between the train split, development split, and test split as SLIM. This baseline does not use external skills or lifecycle operations, so it isolates the effect of learning to interact with search from the effect of managing a skill bank.

SFT on SearchQA. SFT is evaluated as a supervised fine-tuning baseline on SearchQA (Chung et al., 2024). Training examples are constructed only from the train split using the same response format as the search-enabled environment. The model is optimized to imitate the target reasoning-and-answer trajectory, while validation is used only for checkpoint selection. SFT does not use outcome-reward RL, external skills, lifecycle auditing, or trajectories from the test split during training.

Reject Sampling on SearchQA. Reject Sampling follows the Search-R1 baseline construction (Jin et al., 2025; Ahn et al., 2024). For each training prompt, we sample five search-enabled candidate trajectories with the same search interface and action budget, keep trajectories whose final answer is correct under exact match, and fine-tune the model on the selected trajectories. This preserves the multi-turn LLM–search interaction format while replacing online RL with filtered supervised learning. Validation is used for checkpoint selection, and the test split is held out until reporting.

C Evaluation Setup

Benchmark Protocol. We evaluate all methods on the two agent benchmarks used in the main paper. ALFWorld (Shridhar et al., 2021) is a long-horizon text-interaction benchmark in which the agent must complete household tasks by issuing admissible text actions. We report both the overall success rate and task-type success rates for Pick, Look, Clean, Heat, Cool, and Pick2, following the task categories used by the environment. Although ALFWorld has a fixed set of task types, RL supervision is collected over multi-step action trajectories, so each episode contributes many action-level decisions under the shared rollout protocol. SearchQA follows the search-augmented question-answering setting of Search-R1 (Jin et al., 2025). The agent interacts with a search tool and must eventually output a final answer for questions from NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023), HotpotQA (Yang et al., 2018), 2Wiki (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023). All methods use the same environment-side action parser, search interface, and trajectory termination rule within each benchmark. The cross-backbone experiments additionally evaluate SLIM, SkillRL, and Skill0 on WebShop (Yao et al., 2022), where agents search, inspect, and select products according to user-specified requirements.

Data Splits and Data Usage. We keep the train split, development split, and test split separate. ALFWorld follows the official data-split protocol (Shridhar et al., 2021) for household text-interaction tasks. For SearchQA, the train split draws questions from HotpotQA and NQ. The development split draws from HotpotQA, TriviaQA, 2Wiki, NQ, and MuSiQue. The test split draws from PopQA, 2Wiki, TriviaQA, HotpotQA, NQ, MuSiQue, and Bamboogle. WebShop follows the common protocol used in prior work for assigning product-shopping goals to the train split, development split, and test split. The protocol is fixed and shared across all methods: policy optimization uses the train split, hyperparameters are selected on a smaller subset of the train split, and training-time skill management and validation use the development split. The test split is reserved for final reporting after the policy checkpoint and active skill set are frozen and does not participate in training.

Success Metrics. The primary metric in all tables is success rate. Avg. denotes the overall success rate. In ALFWorld, a rollout succeeds when the environment signals completion of the target objective. In SearchQA, success requires a final answer matching the benchmark ground truth under the shared rule. WebShop reports binary success over goals in its test split.

Robustness Diagnostics. The main results and training curves report the original runs. Appendix D.4 additionally reports three independent full training runs of SLIM, SkillRL, and Skill0 for each Qwen3-4B/8B and ALFWorld/SearchQA combination. We summarize success rate with the mean and standard deviation across runs and report the initial, maximum, minimum, and final active skill counts for each ALFWorld SLIM run. Lifecycle thresholds are selected on a smaller subset of the train split and held fixed across seeds. Bootstrap resampling of test outcomes from the original runs provides a complementary assessment of performance stability.

Checkpoint Selection. SLIM, SkillRL, Skill0, and the other online-RL baselines use the final checkpoint after the fixed training horizon for each benchmark: 120 GRPO steps on ALFWorld, 180 on SearchQA, and 150 on WebShop. Training proceeds for the full horizon, with validation used for lifecycle updates and monitoring. SFT and Reject Sampling use validation-based checkpoint selection according to their respective supervised training protocols.

Final Evaluation Protocol. Final evaluation uses the checkpoint specified above with all training-time adaptation disabled. For SLIM, the final active skill set is fixed before test evaluation; test rollouts may retrieve from this frozen active set, but retain, retire, expand, lifecycle audit, and skill creation are disabled. With-skill and no-skill evaluations use the same trained SLIM policy; the latter disables skill insertion at inference. For Skill0, the curriculum state is fixed by its schedule before final evaluation. For SkillRL, the skill bank and retrieval configuration are fixed before test evaluation. Prompt-based and agent- or memory-based baselines are also evaluated on the same test split; when a method accumulates memory during evaluation, that memory is created only from earlier evaluation episodes of that method and is not shared across methods.

Fairness Controls. All main comparisons use Qwen3-4B (Yang et al., 2025) as the backbone model. Online-RL methods start from the same base checkpoint and share the benchmark environment, reward, action format, training horizon, and evaluation protocol. Prompt-based, agent, and memory baselines use the same served backbone and environment wrappers, including the action parser and success criterion. We align prompt and response limits, rollout horizons, validation cadence, and test splits where applicable, while retaining each method's validated configuration. In particular, SkillRL uses top-6 retrieval and reference-policy KL regularization, whereas SLIM uses top-3 retrieval without actor-side KL. Within-SLIM ablations hold the other training settings fixed to assess the lifecycle components. For methods with skill creation or bank updates, comparable modules use the same underlying components where applicable. With-skill methods share the initial SkillRL-style bank family, the insertion format in Figure A6, and the embedding backbone, though retrieval caps remain method-specific. SLIM and SkillRL use the same OpenAI-compatible `o3` skill creator under their respective output formats; EvolveR and memory baselines use the same service-model family where applicable. Methods without skill creation receive no generated skills, and no method creates or updates skills from the test split. Both RL and service-based evaluation use SGLang with shared default inference settings and sampled decoding unless a method's protocol specifies otherwise.

No Cold-Start SFT. All RL baselines are evaluated under the same controlled no-warmup setting. Specifically, all RL-based methods start from the same base checkpoint without cold-start SFT or warmup, so the comparison isolates online RL and lifecycle behavior rather than supervised pre-adaptation. This avoids confounding lifecycle effects with a policy that has already been trained to follow a particular skill format, trajectory style, or action pattern before RL begins. We select method-specific hyperparameters on a smaller subset of the train split and fix them before the main training runs.

Leakage Prevention. The key separation in our protocol is that training-time control signals are never computed from the test split. In particular, SLIM estimates marginal external contribution, routed failure buckets, retain/retire decisions, and expansion triggers only on the development split, never on tasks from the test split. Expansion prompts are constructed strictly from validation-side audit failures; they never include prompts, trajectories, failures, labels, or answers from the test split. Expanded skills are created before final evaluation, and the skill creator never receives prompts, trajectories, failures, or labels from the test split. During final evaluation, the policy checkpoint

Table A1: Generalization to an additional backbone and benchmark. Each method is independently trained for each backbone and benchmark. The left group reports success rate (%); the right group reports the initial, maximum, minimum, and final active skill counts of the corresponding SLIM run. † denotes inference with external skills. **Best** success rates are highlighted.

Benchmark	Backbone	Success rate (%)			SLIM active skills			
		SkillRL†	Skill0	SLIM†	Init.	Max.	Min.	Final
ALFWorld	Qwen3-4B	75.0	74.2	**87.5**	38	46	21	21
	Qwen3-8B	84.7	82.1	**89.1**	38	40	13	15
SearchQA	Qwen3-4B	38.1	39.3	**41.0**	34	37	4	4
	Qwen3-8B	41.7	41.4	**42.7**	34	36	2	2
WebShop	Qwen3-4B	65.8	61.2	**71.0**	57	63	24	28
	Qwen3-8B	75.6	72.4	**79.8**	57	60	16	19

and active skill set are frozen; test rollouts may retrieve from the frozen active set, but they cannot create, edit, retire, or expand skills. SLIM, Skill0, and SkillRL follow the same separation between the train split, development split, and test split. The development split supports training-time skill management: SLIM audits and updates its active skills, Skill0 updates its curriculum, and SkillRL evolves its skill bank. Examples in the test split are held out until reporting. For online memory baselines, any memory accumulated during final evaluation is private to that baseline and arises only from earlier episodes in the same sequential evaluation run; it is not used to select checkpoints, update model parameters, tune prompts, or modify skill banks before evaluation. This prevents labels, trajectories, and failures from the test split from entering policy optimization, skill-bank updates, lifecycle decisions, or skill expansion.

D Additional Experimental Results

This appendix further examines the generalization, robustness, and practical behavior of SLIM. We study generalization to an additional backbone and benchmark, transfer of the final active skill bank, sensitivity to skill initialization, skill creators, and hyperparameters, and stability across independent training runs. We also provide expanded baseline comparisons and an analysis of audit overhead.

D.1 Generalization to an Additional Backbone and Benchmark

We extend the main evaluation with Qwen3-8B as an additional policy backbone and WebShop (Yao et al., 2022) as an additional benchmark. We independently train SLIM, SkillRL, and Skill0 with both Qwen3-4B and Qwen3-8B on ALFWorld, SearchQA, and WebShop. Table A1 brings together task performance and the active skill counts of each SLIM run, showing how lifecycle management adapts to the policy backbone and task demands. The training and evaluation settings are provided in Appendix B.3.

Performance across Backbones. SLIM achieves the highest success rate. With the additional Qwen3-8B backbone, SLIM reaches 89.1 on ALFWorld and 42.7 on SearchQA, improving over the strongest baseline by 4.4 and 1.0 points, respectively. These results extend the performance advantage of lifecycle management to the larger policy backbone.

Generalization to WebShop. The additional WebShop benchmark tests lifecycle management in web interaction, where agents must search for products, inspect their attributes, and select an item that satisfies the task requirements. SLIM reaches 71.0 with Qwen3-4B and 79.8 with Qwen3-8B, exceeding the strongest baseline by 5.2 and 4.2 points, respectively. This extends the gains to a third environment with a different interaction structure.

Backbone-Dependent Lifecycle Endpoints. The active skill counts in Table A1 show that every setting first expands beyond its initialization and subsequently contracts. Qwen3-8B retains fewer final skills than Qwen3-4B on all three environments. The larger backbone achieves stronger performance with less persistent external support, consistent with the relationship between parametric knowledge

Table A2: Transfer evaluation of the final active skill bank learned by SLIM. "None" uses no external skill, "Initial skill bank" uses the pre-training skill bank, and "Final SLIM active skills" uses the final active set learned by SLIM. The transferred skill bank is fixed and no lifecycle update is performed during evaluation.

Prompting Method	Skill Source	ALFWorld Avg.	SearchQA Avg.
Zero-Shot	None	41.4	32.3
	Initial skill bank	63.3	31.5
	Final SLIM active skills	65.8	32.7
Few-Shot	None	39.1	35.5
	Initial skill bank	64.1	34.8
	Final SLIM active skills	66.9	35.8

Table A3: Sensitivity to the initial skill bank. The table reports final performance and final lifecycle statistics under different initialization conditions. The noisy bank randomly replaces 30% of the original skills one-for-one, keeping 38 initial skills.

Initial Skill Bank	Final Avg.	Final Active Skills	Retired Skills	Expanded Skills
Empty skill bank	76.4	18	8	26
Weak initial skill bank	81.2	23	15	29
Noisy initial skill bank	85.6	25	46	33
Original initial skill bank	87.5	21	33	16

and external support discussed in Appendix A.1. The endpoint also varies by environment. SearchQA converges to an almost empty active set, while ALFWorld and WebShop retain compact sets of procedural skills. Thus, SLIM learns an external skill boundary that adapts to both the policy backbone and the task.

D.2 Transfer of the Final Active Skill Bank

Table A2 transfers different skill sources to policies that did not participate in lifecycle training: no external skills, the initial skill bank before RL, and the final active skill set learned by SLIM. On ALFWorld, the transfer is strong: final SLIM skills improve zero-shot and few-shot policies by 24.4 and 27.8 points over no-skill prompting, and by 2.5 and 2.8 points over the initial skill bank. This suggests that the learned active set captures reusable procedural guidance rather than only serving as a private scaffold for the trained SLIM policy. SearchQA shows a different pattern. The final active skills improve zero-shot and few-shot prompting only slightly over no-skill prompting, by 0.4 and 0.3 points, but they consistently outperform the initial skill bank by 1.2 and 1.0 points. Thus, on SearchQA the final skill bank mainly filters noisy or less useful external guidance rather than providing large direct transfer gains. This supports the broader view that skill-bank transferability depends on whether the benchmark benefits from persistent external procedural support.

D.3 Sensitivity to Skill Initialization

Table A3 compares four initial banks under the same training and audit protocol. The *empty* setting removes all external skills; when no active task-specific skill is retrieved, validation failures enter an uncovered bucket indexed by task type. This bucket is not part of $\mathcal{A}_t$, but it can trigger the same expansion operator after accumulating n_{expand} failures with no-skill performance below τ_{expand}, thereby bootstrapping the first skills when $\mathcal{A}_0 = \varnothing$. Later expansions use ordinary skill-routed failure buckets. The *weak* bank retains a stratified 25% subset of the original skills. The *noisy* bank randomly selects 30% of the original skills and replaces them one-for-one with mismatched skills, preserving the original bank size of 38. The *original* bank uses the default initialization from the main experiment.

Expanded Skills counts distinct skill artifacts accepted into the bank over the full training run, including those later retired. Accordingly, the cumulative number of retired skills can exceed the initial bank size because it may include expanded skills that are subsequently retired. The budget

Table A4: Performance over three independent full training runs. Entries are mean ± standard deviation of success rate (%). † denotes inference with external skills. **Best** results for each backbone and benchmark are highlighted.

Backbone	Method	ALFWorld	SearchQA
Qwen3-4B	Skill0	73.44 ± 0.78	39.39 ± 0.19
	SkillRL†	76.30 ± 1.63	38.15 ± 0.24
	SLIM†	$\mathbf{88.02 \pm 1.63}$	$\mathbf{41.37 \pm 0.39}$
Qwen3-8B	Skill0	82.55 ± 0.45	41.45 ± 0.14
	SkillRL†	85.68 ± 1.19	41.85 ± 0.17
	SLIM†	$\mathbf{89.32 \pm 1.19}$	$\mathbf{42.70 \pm 0.22}$

Table A5: ALFWorld skill lifecycle statistics of SLIM over three independent training runs for each backbone. Init., Max., Min., and Final denote the initial, maximum, minimum, and final active skill counts.

Backbone	Run	Init.	Max.	Min.	Final
Qwen3-4B	1	38	46	21	21
	2	38	53	17	20
	3	38	43	19	19
Qwen3-8B	1	38	40	13	15
	2	38	48	12	14
	3	38	45	11	15

$B = 2$ applies to each qualifying audited failure bucket rather than globally to an audit cycle. ALFWorld considers at most $M = 4$ task-specific skill-routed or uncovered buckets per audit, so it can accept at most $MB = 8$ new skills in one audit.

Robustness to Initialization. This experiment varies the initial skill bank while keeping the training and audit protocol fixed. Table A3 shows that SLIM is robust to imperfect initial banks because expansion can repair missing coverage and retirement can filter noisy or low-value skills. At the same time, the original initial bank still gives the highest final score, indicating that lifecycle management complements rather than replaces skill initialization.

Expansion from Weak Coverage. The empty and weak settings reach 76.4% and 81.2% while expanding 26 and 29 skills, respectively. Thus, SLIM can construct external support from persistent failures and recover part of the missing task coverage. The remaining gaps to the original setting show that expansion does not fully replace a reasonably informative initial library.

Filtering Noisy Skills. The noisy setting reaches 85.6%, only 1.9 points below the original setting. Starting from 38 skills, it expands 33 and cumulatively retires 46, ending with 25 active skills. This indicates that SLIM removes harmful external knowledge and repairs missing coverage rather than blindly preserving the initial bank.

Initialization Still Matters. The original skill bank gives the best result, but SLIM still substantially reshapes it by retiring 33 skills and expanding 16 new ones, leaving a compact active set of 21 skills. Thus, the gain does not come from static reuse of the initial skills. A reasonable initial bank improves the ceiling, while lifecycle management determines which skills should remain active after RL.

D.4 Robustness of SLIM Performance

We examine robustness through independent training runs, the resulting skill lifecycle trajectories, and bootstrap resampling of test outcomes. Together, these experiments characterize the stability of both task performance and the learned external skill boundary.

Performance across Training Runs. We conduct three independent full training runs for SLIM, SkillRL, and Skill0 with Qwen3-4B and Qwen3-8B on ALFWorld and SearchQA. Table A4 reports the mean and standard deviation across these runs; Table 1 reports the original run. SLIM achieves the

Table A6: Bootstrap robustness of SLIM gains over the strongest skill-based baselines. Gaps are success-rate percentage points. Confidence intervals are computed with 10,000 independent aggregate bootstrap resamples over test outcomes from original runs.

Benchmark	Comparison	Mean Gap	95% CI	Crosses 0?
ALFWorld	SLIM† – Skill0	+13.3	[+3.9, +22.7]	No
ALFWorld	SLIM† – SkillRL†	+12.6	[+3.1, +21.9]	No
SearchQA	SLIM† – Skill0	+1.72	[+1.13, +2.31]	No
SearchQA	SLIM† – SkillRL†	+2.95	[+2.35, +3.52]	No

Table A7: Sensitivity to the skill creator with Qwen3-4B as the policy backbone. Each row reports the success rate (%) of a full SLIM training run with the specified creator and inference with the final active skills.

Skill Creator	ALFWorld	SearchQA
Frozen Qwen3-4B	86.7	41.1
DeepSeek-V4-Flash	88.3	40.1
GPT-4o	85.9	40.9
o3	87.5	41.0

highest mean success rate in every setting. On ALFWorld, it reaches 88.02 ± 1.63 with Qwen3-4B and 89.32 ± 1.19 with Qwen3-8B. On SearchQA, it reaches 41.37 ± 0.39 and 42.70 ± 0.22, exceeding the strongest baseline by 1.98 and 0.85 points, respectively. The gains persist across independent runs and both backbones, showing that lifecycle-guided training consistently improves performance in the two environments.

Stability of Lifecycle Endpoints. Table A5 summarizes the ALFWorld active-set trajectory of each SLIM run. Stochastic task order and rollouts produce different intermediate maxima and minima, while all runs follow the same qualitative pattern of expansion and retirement. Qwen3-4B finishes with 19–21 active skills, and Qwen3-8B finishes with 14–15. The compact endpoints remain stable across runs even as the intermediate trajectories vary. Together with Table A1, this shows that the learned boundary is reproducible within a backbone and adapts across backbones.

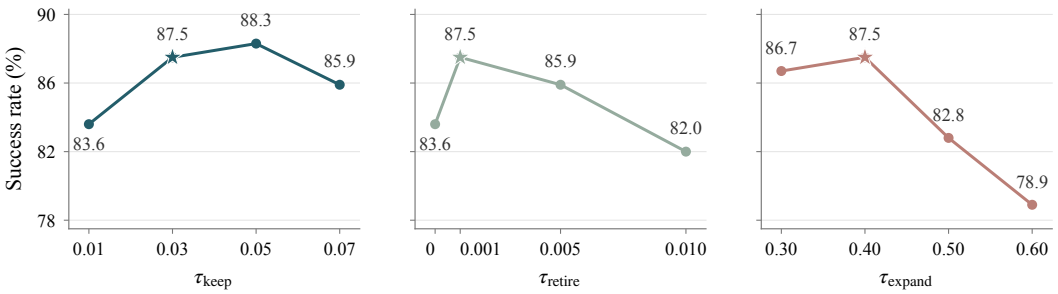
Robustness over Test Outcomes. Table A6 complements the training-run analysis with an independent aggregate bootstrap over test outcomes from the original runs. All confidence intervals for the performance gaps remain above zero. On ALFWorld, the intervals are wider, but the lower bounds remain positive against both Skill0 and SkillRL†. On SearchQA, the intervals are tighter. The bootstrap results and repeated training runs consistently support the performance advantage of SLIM.

D.5 Sensitivity to the Skill Creator

We use o3 in the main experiments to match SkillRL's default skill-creator configuration. To examine how the choice of creator affects SLIM, we fix the policy backbone to Qwen3-4B and replace o3 with frozen Qwen3-4B, DeepSeek-V4-Flash, and GPT-4o while keeping the skill-creation prompt, lifecycle rules, and training settings fixed. Table A7 reports the results on ALFWorld and SearchQA.

Performance across Skill Creators. Changing the skill creator has only a modest effect on SLIM. Success rates range from 85.9 to 88.3 on ALFWorld and from 40.1 to 41.1 on SearchQA across the four creators. Replacing o3 with a frozen Qwen3-4B creator changes the scores by only -0.8 and $+0.1$ points, respectively, reaching 86.7 and 41.1. On ALFWorld, this setting exceeds SkillRL† with o3 by 11.7 points. These results show that SLIM preserves its performance advantage across different skill creators, including one with the same backbone as the policy.

Contribution of Skill Expansion. The no-expansion variant in Table 2 retains MEC-driven retention and retirement and reaches 78.9 on ALFWorld without creator calls, exceeding SkillRL† by 3.9 points and Skill0 by 4.7 points. Enabling expansion adds 7.8 points with a frozen Qwen3-4B creator and 8.6 points with o3. Together, the ablation and creator substitutions attribute SLIM's gains to lifecycle management and subsequent policy adaptation, rather than distillation from o3.



(a) Retention (b) Retirement (c) Expansion

Figure A1: Sensitivity to hyperparameters on ALFWorld. Each panel varies one lifecycle threshold while fixing the other hyperparameters. Stars mark the default values: $\tau_{\text{keep}} = 0.03$, $\tau_{\text{retire}} = 0.001$, and $\tau_{\text{expand}} = 0.40$. All panels use the same success-rate scale.

Table A8: Expanded ALFWorld comparison. All entries report task success rate. † denotes evaluation with retrieved external skills. * denotes closed-source model results copied from SkillRL (Xia et al., 2026). Avg. denotes the overall success rate. **Best** results are highlighted.

Method	Pick	Look	Clean	Heat	Cool	Pick2	Avg.
GPT-4o*	75.3	60.8	31.2	56.7	21.6	49.8	48.2
Gemini-2.5-Pro*	92.8	63.3	62.1	69.0	26.6	58.7	64.3
SimpleMem	**100.0**	25.0	9.5	53.3	39.1	11.8	45.0
MemRL	**100.0**	33.3	10.0	46.2	9.5	6.7	38.6
RLOO	85.3	**100.0**	50.0	75.0	52.2	38.9	62.8
SLIM†	92.9	**100.0**	**91.4**	**78.3**	**88.5**	**81.2**	**87.5**

D.6 Sensitivity to Hyperparameters

We study the three hyperparameters that govern retain, retire, and expand decisions on ALFWorld. We vary one threshold at a time and fix the others at their defaults: $\tau_{\text{keep}} = 0.03$, $\tau_{\text{retire}} = 0.001$, and $\tau_{\text{expand}} = 0.40$. Other settings remain fixed. Figure A1 shows the resulting success rates.

Retention Threshold. Setting τ_{keep} between 0.03 and 0.07 yields 85.9–88.3, with the highest score at 0.05. Lowering it to 0.01 gives 83.6. A smaller threshold allows weakly positive skills to satisfy the retain condition, making retention more permissive. The stronger results at intermediate values support selective retention of skills with clear external value.

Retirement Threshold. The intermediate values 0.001 and 0.005 yield 87.5 and 85.9. Setting $\tau_{\text{retire}} = 0$ makes retirement more conservative and lowers success to 83.6, while increasing it to 0.010 makes retirement more aggressive and lowers success to 82.0. The two-sided decrease reflects the balance between removing redundant support and preserving skills with positive external value.

Expansion Threshold. Values of 0.30 and 0.40 yield 86.7 and 87.5. Increasing τ_{expand} to 0.50 and 0.60 makes more routed regions eligible for expansion and lowers success to 82.8 and 78.9. Focused expansion therefore provides more effective capability coverage. Across the three thresholds, SLIM maintains strong performance over useful operating ranges: $\tau_{\text{keep}} \in [0.03, 0.07]$, $\tau_{\text{retire}} \in [0.001, 0.005]$, and $\tau_{\text{expand}} \in [0.30, 0.40]$.

D.7 Expanded Results

Additional ALFWorld Baselines. The expanded ALFWorld comparison adds closed-source, memory-based, and RL-based baselines beyond the main table. SLIM remains competitive against these broader alternatives while using the same environment and evaluation protocol, which strengthens the conclusion that lifecycle management improves procedural agent training rather than only outperforming a narrow baseline set.

Table A9: Expanded SearchQA comparison. All entries report success rate. † denotes evaluation with retrieved external skills. Avg. denotes the overall success rate. **Best** results are highlighted.

Method	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Avg.
RAG	37.95	58.62	**44.99**	**37.10**	25.40	7.57	21.60	39.78
Search-o1	30.60	51.80	22.60	28.20	38.20	11.20	**41.60**	33.66
Search-R1	37.56	51.05	38.36	33.68	**40.09**	**13.20**	35.20	39.65
SFT	**48.70**	49.40	43.60	28.80	31.40	7.10	32.00	38.40
Reject Sampling	37.98	48.36	38.36	30.83	30.98	9.97	37.60	36.32
SLIM†	38.40	**62.10**	40.40	36.90	31.50	12.70	36.00	**41.04**

Table A10: Audit overhead comparison on the ALFWorld training setting. V denotes one ordinary validation pass, S is the full skill-bank size, S_k is the task-specific active pool for task type k, and K is the bounded SLIM audit budget.

Method	Additional validation-equivalent calls	Skill-bank scaling	Bottleneck	Main benefit
Skill0	$O(1)$ coarse curriculum comparisons	Shrinking skill set	Curriculum validation	Efficient withdrawal toward zero-skill inference
SkillRL	$O(1)$ ordinary validation	$O(S)$ retrieval / maintenance as skills accumulate	Growing skill bank and prompt management	Persistent skill augmentation and exploration support
SLIM	$O(1+K), K \leq 5$	$O(S_k)$ retrieval and bounded audit	Periodic leave-one-skill-out audit	Dynamic external boundary through retain, retire, and expand

Additional SearchQA Baselines. The expanded SearchQA comparison includes retrieval-augmented, supervised, rejection-sampling, and RL-style baselines. These results show that SLIM is not merely benefiting from a stronger prompting or search interface; its gains remain consistent when compared with alternative ways of improving search-augmented QA behavior.

D.8 Audit Overhead Comparison

SLIM introduces leave-one-skill-out audit calls, but the audit budget is periodic and capped. Table A10 shows that SLIM is more expensive than ordinary validation, but its lifecycle cost is bounded by K rather than the full skill-bank size. In contrast, SkillRL has lighter validation but can accumulate a growing skill bank, while Skill0 is cheaper but mainly models progressive withdrawal. We quantify the operational footprint through active skill counts, audit calls, and wall-clock time. SLIM reduces the final active skill count from 38 initial skills plus 16 expansions to 21 active skills, while SkillRL grows to 73. Its audit budget is capped and does not scale with the full skill bank, and wall-clock remains comparable to SkillRL and Skill0. Together with the w/o Retirement and Fixed Active Set Size ablations, these results indicate that SLIM's gains are not due to unbounded external support or simple prompt-budget control.

E Prompts

This section provides the prompt templates used for environment interaction, skill insertion, baseline agents, and skill expansion. We keep placeholders such as `{task_description}` and `{retrieved_memories}` explicit so that the templates can be mapped directly to the training and evaluation protocol.

Environment Interaction Prompts. Figure A3 and Figure A5 show the skill-conditioned rollout prompts used for ALFWorld and SearchQA. Figure A2 and Figure A4 show the corresponding no-skill templates.

Skill Insertion Format. Figure A6 shows how retrieved general and task-specific skills are inserted into the agent context.

Baseline Prompts. Figure A7, Figure A8, Figure A9, Figure A10, and Figure A11 provide the prompts used by prompt-based and memory-style baselines.

Skill Creation Prompt. Figure A12 gives the expansion prompt used by SLIM to create new standalone task-specific `SKILL.md` artifacts from routed failures.

F SKILL BANK DETAILS

Tables A11 and A12 summarize representative skills from the hierarchical skill banks used by SLIM on ALFWorld and SearchQA, respectively. They include general and task-specific skills with concise trigger conditions and one-line procedural content. Dynamically expanded skills are inserted into the task-specific pool of the corresponding task type and become eligible for later retrieval and lifecycle auditing.

G LIMITATIONS

SLIM has three main limitations. First, marginal external contribution is a local single-skill leave-one-out estimate conditioned on the current policy, routing behavior, and active set. It is not a global Shapley-style attribution and does not capture high-order interactions among skills. Second, lifecycle thresholds and audit budgets require tuning on the train split, so transferring the same configuration to substantially different domains may require additional calibration. Third, lifecycle auditing remains practical in our current setting but may become expensive for very large skill banks, where more scalable audit candidate selection would be needed.

Title: ALFWorld No-Skill Rollout Prompt

```
You are an expert agent operating in the ALFRED Embodied
Environment.  Your task is to:  {task_description}
Prior to this step, you have already taken {step_count} step(s).
Below are the most recent {history_length} observations and the
corresponding actions you took:  {action_history}
You are now at step {current_step} and your current observation is:
{current_observation}
Your admissible actions of the current situation are:
[{admissible_actions}].
Now it's your turn to take an action.  You should first reason
step-by-step within <think> </think> tags, then choose one
admissible action within <action> </action> tags.
```

Figure A2: ALFWorld no-skill rollout prompt.

Title: ALFWorld Skill-Conditioned Rollout Prompt

```
You are an expert agent operating in the ALFRED Embodied
Environment.  Your task is to:  {task_description}
{retrieved_memories}
Prior to this step, you have already taken {step_count} step(s).
Below are the most recent {history_length} observations and the
corresponding actions you took:  {action_history}
You are now at step {current_step} and your current observation is:
{current_observation}
Your admissible actions of the current situation are:
[{admissible_actions}].
Now it's your turn to take an action.  You should first reason
step-by-step within <think> </think> tags, then choose one
admissible action within <action> </action> tags.
```

Figure A3: ALFWorld skill-conditioned rollout prompt. The placeholder `{retrieved_memories}` is filled by the skill insertion format in Figure A6.

Title: SearchQA No-Skill Rollout Prompt

```
You are an expert agent tasked with answering the given question
step-by-step.
Your question:  {task_description}
Prior to this step, you have already taken {step_count} step(s).
History:  {memory_context}
Now it's your turn to respond.  First reason within <think> </think>
tags.  Then choose one action:  search using <search> your query
</search>, or answer using <answer> final answer </answer>.
```

Figure A4: SearchQA no-skill rollout prompt.

Title: SearchQA Skill-Conditioned Rollout Prompt

```
You are an expert agent tasked with answering the given question
step-by-step.
Your question: {task_description}
{retrieved_memories}
Prior to this step, you have already taken {step_count} step(s).
History: {memory_context}
Now it's your turn to respond. First reason within <think> </think>
tags. Then choose one action: search using <search> your query
</search>, or answer using <answer> final answer </answer>.
```

Figure A5: SearchQA skill-conditioned rollout prompt.

Title: Retrieved Skill Insertion Format

```
### GENERAL SKILLS ###
Here are some useful general strategies for solving the task
effectively:
1. {title}: {principle} Apply this when {when_to_apply}
...
### TASK: {task_type} ###
For {readable_task} tasks, apply these specific strategies:
1. {title}: {principle} Apply this when {when_to_apply}
...
```

Figure A6: Skill insertion format used by SLIM and skill-conditioned baselines. General skills are inserted as a group when active, while task-specific skills are retrieved by task type and semantic similarity.

Title: Zero-Shot and Few-Shot Prompting

```
Zero-shot:
{obs_text}
Few-shot prefix:
Below are solved examples. Follow the same output format and
decision style.
[Example 1] User: {example_user} Assistant: {example_assistant}
[Example 2] User: {example_user} Assistant: {example_assistant}
[Current Task]
{obs_text}
```

Figure A7: Zero-shot and few-shot prompting templates. Skill-conditioned variants insert the retrieved skill block from Figure A6 into the current task context.

Title: ReAct Prompt

```
System: You are following the ReAct paradigm: interleave
explicit reasoning with environment actions. Reason concisely but
concretely. Always output reasoning inside <think>...</think> and
the single next action inside the required action tags.
User: {obs_text}
```

Figure A8: ReAct prompt used under the shared environment protocol.

Title: Reflexion Prompt

```
System: You are using the Reflexion style. Before acting, consult
prior reflective notes about common mistakes. Avoid repeating
known errors. Keep the output format exact.
User: Past reflections from related mistakes:
- {reflection_1}
- {reflection_2}
{obs_text}
Reflection generation: Write one short actionable reflection that
would help avoid the same failure next time.
```

Figure A9: Reflexion prompt and reflection-generation instruction.

Title: ExpeL Prompt

```
System: You are using the ExpeL style: consult reusable lessons
distilled from previous trajectories, then act using the current
environment state.
User: Retrieved experience lessons:
- {lesson_1}
- {lesson_2}
{obs_text}
Lesson distillation: Distill one compact reusable lesson from this
trajectory. It should be transferable and phrased as an actionable
rule.
```

Figure A10: ExpeL prompt and lesson-distillation instruction.

Title: Mem0 Prompt

```
System: You are using a Mem0-style external memory. Retrieve
small factual or procedural memories from prior tasks and use them
only when relevant.
User: Relevant memories from prior tasks:
- {memory_1}
- {memory_2}
{obs_text}
Memory extraction: Extract up to 3 short atomic memories from this
trajectory and return them as a JSON list of strings.
```

Figure A11: Mem0 memory prompt and memory-extraction instruction.

Title: SLIM Skill Creation Prompt

```
System: Create one new standalone task-specific SKILL.md artifact
that addresses the dominant failure pattern. Do not rewrite an
existing skill and do not create a general or foundational skill.
User: Task type: {task_type}
Failed tasks and traces: {failure_bucket}
Currently selected skills that were insufficient: {active_skills}
Output a skill with:
1. Skill name
2. Routing-oriented description
3. Trigger condition
4. Concise procedural workflow
Reject generic names, duplicate titles, overly short workflows, and
any output that targets general skills.
```

Figure A12: Skill creation prompt used by SLIM during expansion. The prompt follows an Anthropic-style skill-creator workflow and produces new task-specific `SKILL.md` artifacts.

Table A11: Representative ALFWorld skills used by SLIM, including the skills analyzed in the lifecycle case study.

Task	Skill ID	Skill Name	Trigger	Content
General	gen_004	Track Counts & Progress	Multi-instance goals such as putting two objects.	Maintain a counter of remaining goal objects and stop only when the count reaches zero.
General	gen_011	Efficient Relation Search	Goals mentioning both a target and reference object.	Search one object near the other instead of treating them independently.
General	gen_001	Systematic Exploration	Goal object count is unmet and unexplored locations remain.	Search plausible surfaces or containers once before revisiting checked locations.
General	gen_002	Immediate Acquisition	First visual confirmation of a goal-relevant object.	Take a required object as soon as it becomes visible and reachable.
General	gen_003	Destination First Policy	Holding a goal object after identifying its target location.	Navigate directly to the receptacle and place the object before resuming search.
pick_and_place	pic_001	Systematic First-Pass Search	Before acquiring required objects.	Maintain a checklist of visible and closed candidates and inspect each once.
pick_and_place	pic_002	Grab When Seen	First sight of an unheld target object.	Immediately take a needed visible object before moving elsewhere.
clean	cle_003	Sink First for Cleaning	Target object is held and must be clean.	Go to the nearest sink or basin and clean the object before placement.
clean	cle_005	State Verification Before Drop	After cleaning and before final placement.	Verify the object is clean; clean again if the state is uncertain.
heat	hea_003	Open Then Heat	At the microwave with the target object held.	Open the microwave, place the object inside, and execute the heat action.
cool	coo_005	Direct Post-Cooling Delivery	Cooling action succeeds.	Deliver the cooled object directly to the destination without detours.
cool	coo_002	Confirm Object Match	After acquiring an object in a cooling task.	Verify that the held item matches the requested target type; otherwise drop it and resume search.
cool	coo_004	Enforce Cooling Before Placement	Holding the correct object before final placement.	Do not place the target object until a fridge or freezer cooling action has succeeded.
cool	dyn_verify _cooling _completion	Verify Cooling Completion	After placing an object inside a cooling appliance.	Confirm the object is cool before retrieval, then proceed directly to delivery.

Table A12: Representative SearchQA skills used by SLIM.

Task	Skill ID	Skill Name	Trigger	Content
General	gen_001	Decompose Then Search	Complex or multi-part questions.	Break the question into minimal sub-questions and search each before synthesis.
General	gen_002	Precision Query Crafting	Initial query formulation.	Use exact entity names and target attributes while avoiding filler words.
General	gen_003	Iterative Query Refinement	First result set lacks definitive evidence.	Add qualifiers, alternate names, dates, or context instead of repeating the same query.
General	gen_004	Source-Backed Assertions	Before committing to a final answer.	Answer only after locating supporting evidence.
General	gen_005	Cross-Check Multiple Sources	Facts may be outdated or disputed.	Validate names, dates, and numbers against multiple independent sources.
direct_retrieval	dir_001	Isolate Core Query	Start of direct retrieval.	Strip the question to the key entity and sought fact, then search that pair.
direct_retrieval	dir_002	Refine When Empty	Initial search gives weak or no hits.	Reformulate with synonyms, alternate names, dates, or quoted phrases.
direct_retrieval	dir_003	Anchor With Quotes	Distinctive phrases, lyrics, titles, or quotes.	Wrap unique phrases in quotation marks to retrieve exact-match sources.
direct_retrieval	dyn_direct_retrieval _search_first _answer_second	Search First, Answer Second	Factoid or definition queries.	Always run an external search before relying on memory.
multi_hop_reasoning	mul_001	Decompose Question First	Multi-hop questions linking multiple facts.	Split the question into explicit sub-questions before searching.
multi_hop_reasoning	mul_003	Collect-Then-Compare	Comparative multi-hop tasks.	Retrieve concrete values for all items before comparing.
entity_attribute_lookup	ent_001	Direct Attribute Query	Full, unambiguous entity name is given.	Include the full entity name and target attribute in the first search.
entity_attribute_lookup	ent_003	Two-Source Cross-Check	First plausible answer appears or attribute is uncertain.	Confirm the attribute in at least two independent sources.
compare	com_001	Decompose & Isolate	Reading a comparison-type question.	Split the question into entities and the single attribute to compare.
compare	com_003	Normalize Before Comparing	After gathering each entity's attribute.	Convert values to a common comparable form before judging equality or order.