CONTINUAL-SKILL: UTILITY OPTIMIZATION-DRIVEN CO-EVOLUTION OF POLICIES AND SKILL LIBRARIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-improving agents continually accumulate reusable skills, yet they lack an efficient mechanism for determining whether a stored skill still benefits the current policy. Semantic relevance does not reveal a skills actual effect on task performance, while explicit with/without-skill evaluation requires costly additional environment interactions. We introduce **CONTINUAL-SKILL**, a framework that turns ordinary training trajectories into online evidence of skill utility. Its key idea is to randomize skill exposure during training, creating exposed and unexposed observations for candidate skills, and to use inverse-probability weighting to estimate their marginal effects on agent performance. The resulting utility estimates are updated alongside the policy and directly govern skill routing and lifecycle decisions, including promotion, suspension, and archival; persistent task-domain failures additionally trigger new-skill generation. This design allows policy learning and skill reassessment to share the same interaction data, avoiding separate per-skill evaluation rollouts. On ALFWorld and WebShop, CONTINUAL-SKILL remains robust under increasingly interfered skill banks and improves success rate by up to 40.4 percentage points over strong skill-learning baselines on expanded WebShop banks. Results suggest that continual agent improvement requires treating skill utility as a policy-dependent quantity that must be repeatedly identified and updated throughout learning.

1 INTRODUCTION

Language agents increasingly reuse experience through external skill libraries, where successful procedures are distilled into reusable instructions and retrieved for future tasks. This mechanism is appealing for long-horizon self-improvement because it allows an agent to accumulate capabilities without relearning every behavior from scratch. However, the value of stored experience does not remain fixed. As the policy improves and task contexts change, a previously useful skill may become redundant, overly specialized, or even interfere with behavior. Thus, the central challenge is not only how to acquire more skills, but how to keep the skill library useful as the agent itself evolves.

Existing methods mainly address skill construction, retrieval, and internalization, but offer limited mechanisms for continuously reassessing skills already in the library. As illustrated in Figure 1, this creates three coupled challenges: growing skill banks accumulate redundant or conflicting guidance; the utility of a skill changes as the policy improves; and directly measuring this utility through separate with/without-skill audits requires additional environment interactions. Semantic relevance alone therefore cannot determine whether a skill actually benefits the current agent, while explicit auditing becomes increasingly costly as the library grows. This motivates a central question: *can the trajectories already collected for policy learning also reveal which skills remain useful?*

We introduce **CONTINUAL-SKILL**, a framework for continual skill evolution via utility optimization. CONTINUAL-SKILL injects controlled stochasticity into skill exposure during training, so that ordinary agent trajectories naturally contain both exposed and unexposed observations for candidate skills. An inverse-probability estimator then recovers each skill's marginal contribution from task outcomes, allowing the same rollout data to support both policy learning and skill evaluation without separate per-skill environment execution. The resulting utility estimates are used not only for routing, but also for lifecycle management: useful skills are retained or promoted, persistently

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

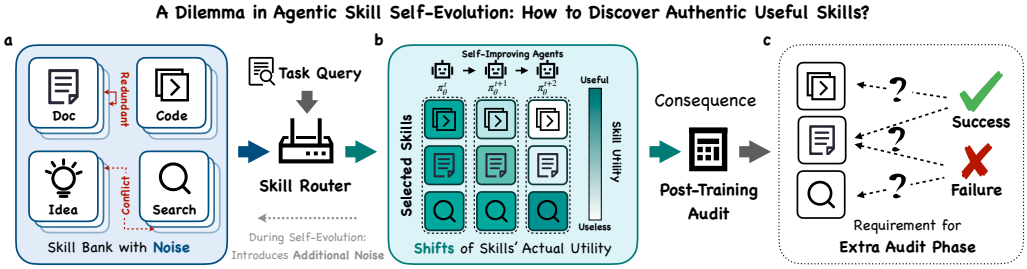


Figure 1: **The dilemma of evolving agent skills.** (a) Retrieval encounters redundant and conflicting knowledge. (b) The usefulness of selected skills changes as the policy improves. (c) Evaluating each skill through a separate audit requires additional interactions. The challenge is to obtain timely evidence of utility during the training process itself.

harmful skills are suspended or archived, and repeated failures can trigger the generation and trial of new skills. Together, these mechanisms couple policy learning with skill evolution.

CONTINUAL-SKILL was evaluated with Qwen2.5-7B-Instruct on ALFWorld and WebShop, using progressively expanded skill banks containing redundant, overspecialized, outdated, mismatched, and conflicting guidance. As skill-bank interference increases, existing baselines degrade substantially, while CONTINUAL-SKILL maintains stronger performance, with gains of up to 40.4 percentage points on expanded WebShop banks. Together, these results support a simple principle for self-improving agents: external skills should be treated as an evolving resource whose usefulness is continually measured and updated alongside the policy.

2 PRELIMINARIES: SKILL-ASSISTED AGENTIC SYSTEMS

Let $x \sim \mathcal{D}$ denote a task sampled from task distribution $\mathcal{D}$. A language policy π_θ, parameterized by θ, interacts with the environment and produces a trajectory τ, which receives an environment reward $r^{\text{env}}(\tau)$. The agent is augmented with a skill bank. Each skill is represented as

$$s = (\text{id}, d, c, m),$$

where id is a unique identifier, d is its task domain, c is the procedural content, and m denotes metadata. At training step t, the skill bank is

$$B_t = B_t^{\text{general}} \cup \bigcup_{g \in \mathcal{G}} B_t^g, \tag{1}$$

where $\mathcal{G}$ is the set of task groups, B_t^{general} contains general skills, and B_t^g contains skills specific to group g. For task x, the retriever selects candidate skills $C(x) \subseteq B_t$, and the router μ chooses a subset $S \subseteq C(x)$ for the agent. Conditioned on S, the policy samples each action as $a_\ell \sim \pi_\theta(\cdot \mid h_\ell, S)$, where h_ℓ denotes the interaction history before step ℓ. The overall objective is

$$J(\theta, B, \mu) = \mathbb{E}_{x \sim \mathcal{D},\, S \sim \mu(\cdot|x,B),\, \tau \sim \pi_\theta(\cdot|x,S)} \left[r^{\text{env}}(\tau) \right]. \tag{2}$$

Thus, task performance depends jointly on the policy π_θ, the skill bank B, and the routing policy μ. CONTINUAL-SKILL treats the skill bank as an evolving state and updates it using utility evidence collected during training.

3 CONTINUAL-SKILL: CONNECTING POLICY LEARNING AND SKILL EVOLUTION

3.1 OVERVIEW: ONE ROLLOUT, TWO LEARNING SIGNALS

CONTINUAL-SKILL couples policy learning with skill evolution through the same on-policy trajectories. At the beginning of each training batch t, we freeze the policy π_{θ_t}, the skill bank, routing

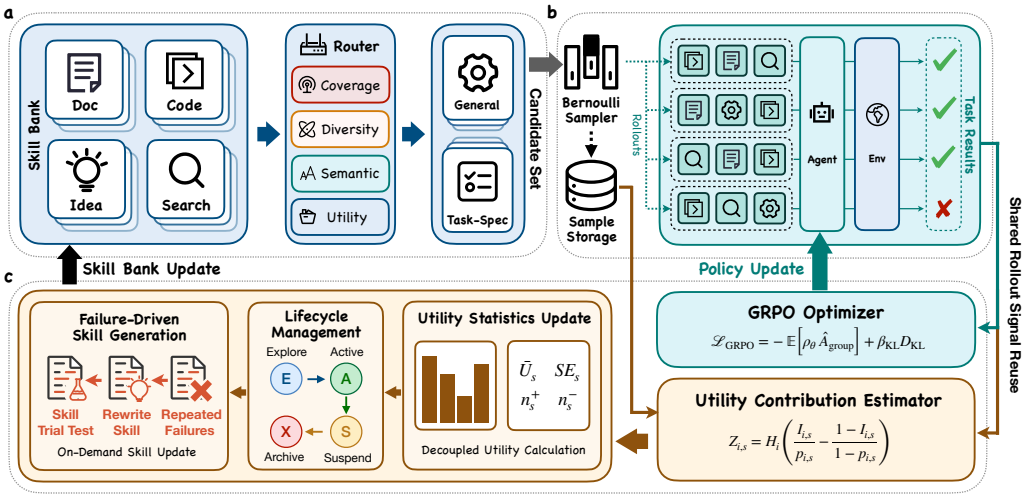


Figure 2: **The CONTINUAL-SKILL training loop.** (a) Semantic relevance, utility, coverage, and diversity determine candidate skills. (b) Recorded exposure assignments condition ordinary agent rollouts. (c) Trajectory feedback supports policy optimization and utility updates. Utility guides routing and lifecycle transitions, while persistent task-domain failures trigger skill generation. The optimizer box summarizes the policy-gradient and KL components; the clipped optimization objective is given in Appendix E.2.

statistics, and historical utility baselines. For each task, the router first retrieves a set of candidate skills and randomly exposes a subset of them to the agent. The resulting trajectories provide two complementary learning signals: environment rewards optimize the policy through GRPO, while skill exposure and task outcomes estimate which skills actually helped. Both updates are applied only after the batch is completed and affect the next batch, as illustrated in Figure 2.

3.2 SKILL EXPOSURE

For the i-th task attempt x_i, the router constructs a candidate set C_i using semantic relevance and recent utility, together with coverage and diversity constraints. For each candidate skill $s \in C_i$, we independently sample whether it is exposed:

$$I_{i,s} \sim \text{Bernoulli}(p_{i,s}), \qquad S_i = \{s \in C_i : I_{i,s} = 1\}, \tag{3}$$

where $I_{i,s} = 1$ means that skill s is provided to the agent, and $p_{i,s} \in (0, 1)$ is its exposure probability.

The key role of the randomization is to create natural treatment and control observations inside ordinary post-training. Across repeated attempts, the same candidate may sometimes be exposed or withheld, allowing its effect to be estimated without explicitly rerunning every trajectory with and without that skill. We record C_i, $I_{i,s}$, $p_{i,s}$, and the corresponding policy and skill versions before interaction so that every outcome can be traced back to the knowledge context that produced it.

3.3 ESTIMATING A SKILL'S MARGINAL UTILITY

Skill Utility. The central question is not only whether a skill is semantically relevant, but whether providing it improves the current agent performance. For trajectory i, we summarize task quality as

$$R_i = w_y y_i - w_f(1 - y_i) - w_\ell \frac{L_i}{L_i^{\max}} - w_v(1 - v_i), \tag{4}$$

where y_i denotes task success, $L_i/L_i^{\max}$ is normalized interaction length, and v_i is the fraction of valid actions. We center this feedback using the historical mean of the corresponding task group g_i:

Algorithm 1 CONTINUAL-SKILL: shared-rollout policy learning and skill governance

Require: Policy π_{θ_0}, bank $\mathcal{B}_0$, tasks $\mathcal{D}$, routing and audit rules
for each training batch t **do**
  Freeze policy, skill contents, routing statistics, and historical baselines.
  for each sampled task and repeated attempt i **do**
    Construct C_i with exploitation, coverage, and diversity selection.
    Assign candidate exposures; ordinary sampling follows Eq. (3).
    Record $(C_i, p_{i,s}, I_{i,s})$ and versions; collect τ_i and r_i^{env}.
  end for
  Compute Eqs. (4)–(6); update candidate moments and arm counts.
  Incorporate this batch into historical task-group feedback statistics.
  Compute GRPO advantages from environment rewards and update θ.
  if an audit is due **then**
    Update lifecycle states; generate and screen eligible failure-driven proposals.
    Register new skills with fresh utility statistics and a five-step cold-start window starting next step.
  end if
  Roll statistical windows as scheduled; refresh indices and save audit records.
end for
Ensure: Updated policy, skill bank, and utility-guided routing state

$$H_i = R_i - b_{g_i,t}. \tag{5}$$

The historical baseline centers feedback using the task group's past mean and is fixed before the current batch. Skill attribution uses R_i; policy optimization uses the environment reward r_i^{env}.

We then assign skill s the contribution

$$Z_{i,s} = H_i \left(\frac{I_{i,s}}{p_{i,s}} - \frac{1 - I_{i,s}}{1 - p_{i,s}} \right). \tag{6}$$

Why this form? The exposed and unexposed observations receive coefficients $1/p_{i,s}$ and $-1/(1-p_{i,s})$, respectively. These inverse-probability terms account for their potentially unequal assignment probabilities. Let $\mathcal{F}_i$ denote the pre-interaction information defined in Appendix B. Under the assumptions of Proposition 1, ordinary randomized exposure identifies

$$\begin{aligned} \mathbb{E}[Z_{i,s} \mid \mathcal{F}_i] &= \mathbb{E}[H_i \mid I_{i,s} = 1, \mathcal{F}_i] - \mathbb{E}[H_i \mid I_{i,s} = 0, \mathcal{F}_i] \\ &= \Delta_{i,s}. \end{aligned}$$

Here, $\Delta_{i,s}$ is the conditional effect of exposing skill s on the feedback in Eq. (4), averaged over the other candidates' randomized assignments. The windowed estimate is

$$\widehat{U}_s = \frac{1}{|\mathcal{W}_s|} \sum_{i \in \mathcal{W}_s} Z_{i,s}, \tag{7}$$

where $\mathcal{W}_s$ contains recent candidate opportunities for skill s. The estimate aggregates exposure effects across the policies, task contexts, and assignment mechanisms represented in the window and guides subsequent routing. Proposition 1 establishes identification for individual candidate opportunities; tracking error under policy and retrieval drift remains an open analytical question.

3.4 POLICY LEARNING WITH GRPO

CONTINUAL-SKILL updates the policy with GRPO(Shao et al., 2024). Environment rewards from repeated attempts of each task define relative advantages for a clipped policy objective with KL regularization. Optimization is performed over generated action tokens; Appendix E.2 gives the objective and hyperparameters.

Policy learning and cost. Agent policy learning uses the standard GRPO algorithm(Shao et al., 2024). It constructs relative advantages within repeated attempts of each task and optimizes a clipped policy objective. Task groups used for skill baselines and repeated-task groups used for policy advantages serve different functions. Algorithm 1 gives the complete CONTINUAL-SKILL procedure; Appendix E.2 provides the objective and update timing. Utility estimation adds $O(NK_C)$ bookkeeping and requires no separate per-skill environment rollouts.

3.5 UTILITY-DRIVEN SKILL EVOLUTION

CONTINUAL-SKILL uses the contributions in Eq. (6) to update both routing and the skill bank. Recent utility guides candidate selection together with semantic relevance, while accumulated evidence supports reversible downweighting of low-value skills. A coverage channel seeks fresh evidence for skills whose observations have become stale (Appendix C).

At scheduled audits, each skill is evaluated using its windowed utility, empirical standard error, and observation counts in both exposure arms. Sufficient negative evidence triggers suspension; repeated negative confirmations lead to archival. Repeated positive evidence supports promotion or recovery, while non-mature skills may be archived when sufficient evidence indicates inadequate benefit. Otherwise, the state is retained. Lifecycle transitions follow utility-based heuristic rules (Appendix D). Archived skills are excluded from retrieval, while their records are retained.

Persistent task-domain failures trigger the generation of a bounded set of new skills from recent trajectories. Accepted proposals enter the bank with fresh utility statistics and receive reserved coverage candidate slots for five training steps, beginning at the next step after registration. Each domain provides up to two coverage slots; excess eligible new skills are selected by random sampling. Their recorded exposure assignments and trajectory feedback contribute to the utility estimator in Eq. (6), and the accumulated evidence guides subsequent routing and lifecycle decisions. Appendix E describes generation, screening, and cold-start scheduling. Routing and lifecycle updates take effect at batch boundaries, so feedback from each batch shapes skill use in subsequent batches.

4 EXPERIMENTAL SETUP

Environments and training. We evaluate Qwen2.5-7B-Instruct(Qwen et al., 2024) on ALFWorld and WebShop: the former requires manipulating objects from textual observations, while the latter requires search and purchasing under multiple user constraints(Shridhar et al., 2021; Yao et al., 2022). Table 2 reports the complete comparison under one fixed seed shared across methods. Training-dynamics and component analyses use three independent seeds to assess run-to-run variability in those settings. Each training run consists of 200 steps with 64 base trajectories per step, in addition to validation, auditing, and skill-management overhead. We evaluate our implementations of Skill1 (Shi et al., 2026), SkillRL (Xia et al., 2026a), SLIM (Shen et al., 2026), and CONTINUAL-SKILL using the shared training configuration in Appendix G. Appendix H provides the environment interaction prompts. We report system-level token costs in Section 5.3, with details in Appendices E.4 and G.

Skill-bank interference. For each environment, we use the initial skill bank from SkillRL (Xia et al., 2026a) as the Small bank, shared across all compared methods. We preserve its task-domain coverage and construct the Medium and Large banks using five perturbations: redundant paraphrases, overspecialized procedures, incomplete or outdated guidance, cross-task mismatches, and plausible conflicts. Original, $5\times$, and $10\times$ banks are denoted Small, Medium, and Large, respectively, and satisfy

$$\mathcal{B}^{\text{Small}} \subset \mathcal{B}^{\text{Medium}} \subset \mathcal{B}^{\text{Large}}. \tag{8}$$

Each added entry retains its parent, provenance, and perturbation label for analysis. Utility estimates are computed from exposure assignments and trajectory feedback. The expanded banks vary both size and perturbation composition to emulate realistic skill-bank noise interference.

Table 1: Initial skill-bank sizes. All scales preserve the base skills.

Environment	Small / Original	Medium / 5×	Large / 10×
ALFWorld	44	220	440
WebShop	54	270	540

Table 2: Final test performance under different skill-bank conditions. ALFWorld reports category-level SR and overall SR; Avg. is the overall SR, computed as the total number of successful tasks divided by all 134 evaluation tasks. WebShop reports SR and Score (both in percent). For CONTINUAL-SKILL, bold and underlining indicate the best and second-best results, respectively, within each bank condition and column, including ties.

		ALFWorld							WebShop	
Bank	Method	Pick	Look	Clean	Heat	Cool	Pick2	Avg.	SR	Score
Small	Skill1	66.67	100.00	77.42	60.87	85.71	82.35	77.61	46.80	52.89
	SkillRL	75.00	83.33	67.74	82.61	90.48	82.35	79.10	44.40	57.40
	SLIM	66.67	94.44	77.42	82.61	80.95	58.82	76.87	53.40	70.07
	CONTINUAL-SKILL	**75.00**	83.33	**83.87**	65.22	85.71	**82.35**	**79.10**	48.60	53.99
Medium	Skill1	75.00	77.78	80.65	82.61	90.48	76.47	80.60	22.20	34.26
	SkillRL	70.83	72.22	64.52	65.22	76.19	17.65	62.69	10.80	28.06
	SLIM	58.33	94.44	54.84	56.52	61.90	41.18	60.45	4.40	16.84
	CONTINUAL-SKILL	**79.17**	**100.00**	70.97	65.22	**90.48**	**94.12**	**81.34**	**53.60**	**62.00**
Large	Skill1	79.17	66.67	80.65	78.26	95.24	76.47	79.85	9.40	26.44
	SkillRL	70.83	66.67	77.42	65.22	80.95	17.65	65.67	8.60	25.03
	SLIM	58.33	77.78	45.16	56.52	57.14	5.88	50.75	7.40	18.37
	CONTINUAL-SKILL	66.67	**88.89**	**83.87**	73.91	85.71	**88.24**	**80.60**	**49.80**	**58.49**

5 RESULTS AND ANALYSIS

5.1 TASK PERFORMANCE UNDER SKILL-BANK INTERFERENCE

CONTINUAL-SKILL achieves the best or second-best overall success rate in all six environment–bank combinations. CONTINUAL-SKILL achieves the best or second-best overall SR in all six combinations of environment and initial skill bank. Its advantage becomes clearer on expanded WebShop banks: it ranks second on Small initial banks, then achieves the highest SR on both Medium and Large initial skill banks.

CONTINUAL-SKILL shows strong robustness under WebShop interference. Expanded skill banks sharply degrade all three baselines on WebShop; for example, Skill1's SR falls from 46.80% with the Small bank to 9.40% with the Large bank (Table 2; Figure 3). However, CONTINUAL-SKILL achieves WebShop success rates of 48.60%, 53.60%, and 49.80% on Small, Medium, and Large banks, respectively. On Medium and Large, it exceeds the strongest compared baseline by 31.40 and 40.40 percentage points.

5.2 TRAINING DYNAMICS AND GOVERNANCE

The performance gap emerges during training. Figure 4 shows training and validation performance in both environments. On ALFWorld, both methods' training SRs improve as training progresses, reaching approximately 80%–90% in the later stages; CONTINUAL-SKILL's validation SR exceeds Skill1's during most of training, both ending at approximately 85%. On WebShop, both methods start with low success rates, and the gap gradually widens: CONTINUAL-SKILL's training SR reaches approximately 60% in the later stages, compared with approximately 25% for Skill1. Validation SR shows the same direction of advantage, ending at approximately 50% and 30%, re-

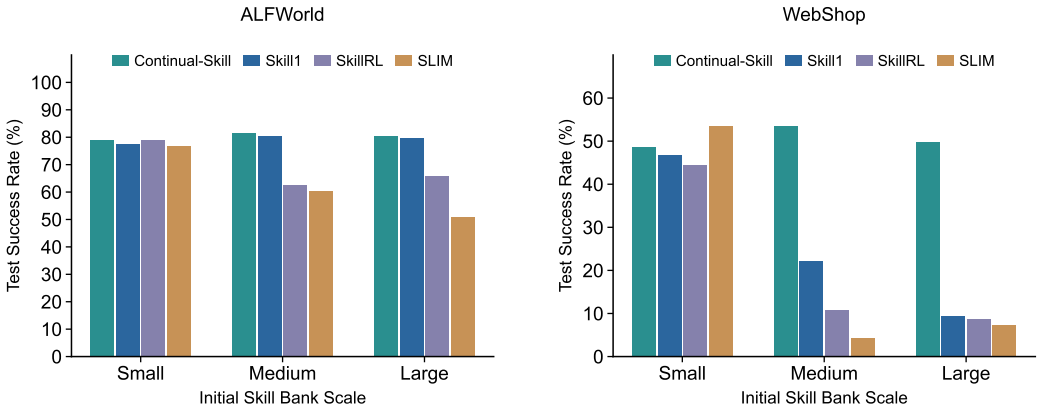


Figure 3: **Final test success rates under different skill-bank conditions.** Baseline performance generally shows a downward trend as skill bank perturbations increase, particularly on WebShop; CONTINUAL-SKILL consistently achieves the best or second-best overall SR throughout and maintains strong robustness under WebShop interference.

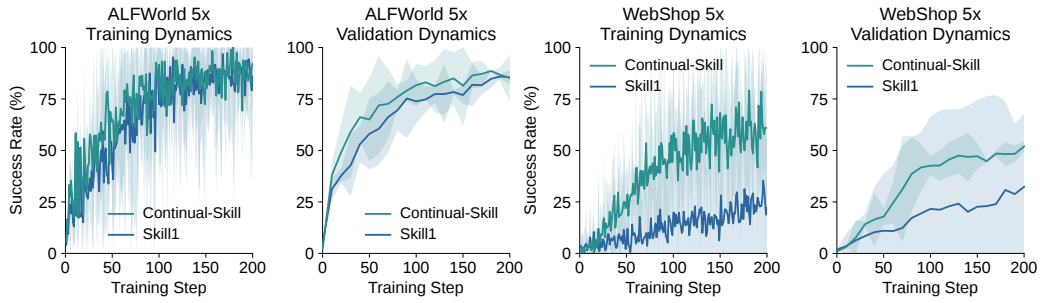


Figure 4: **Training and validation dynamics with Medium ($5\times$) skill banks.** From left to right: ALFWorld training, ALFWorld validation, WebShop training, and WebShop validation on Success Rate, comparing CONTINUAL-SKILL and Skill1 as training progresses. Shading around the validation curves indicates 95% confidence intervals across three different training seeds.

spectively. The curves show improving performance in both environments and a larger training and validation gap on WebShop.

Lifecycle management and generation improve task success. We conduct a separate three-seed paired ablation on Large ALFWorld using the shared training configuration in Appendix G. No-management retains utility estimation and routing while disabling lifecycle transitions and skill generation. Full improves mean test SR from 78.86% to 81.34%, with a mean paired gain of 2.49 percentage points and positive gains across all three seeds (Table 3).

5.3 RESOURCE-INDEXED PERFORMANCE

SLIM assesses skill utility through leave-one-skill-out validation(Shen et al., 2026), whereas CONTINUAL-SKILL reuses ordinary training trajectories. Figure 5 reports training and validation token costs separately. On Large ALFWorld, CONTINUAL-SKILL uses 2.3M versus 3.1M training tokens per step and 5.2M versus 10.3M validation tokens per round, corresponding to reductions of approximately 27.4% and 49.7%, respectively.

Figure 6 evaluates performance against cumulative LLM token consumption. The total sums input and output tokens across training interaction, method-specific skill operations, and periodic validation. Inputs include task instructions, injected skills, interaction history, and current observations;

Table 3: Paired governance comparison on ALFWorld Large (10 $\times$) initial bank. Values are final test SR (%). SD is across three training seeds; differences are computed within seed before aggregation. Full denotes this component experiment's complete configuration.

Configuration	Seed 1	Seed 2	Seed 3	Mean	SD
No-management	78.36	79.10	79.10	78.86	0.43
Full	82.09	80.60	81.34	81.34	0.75
Gain	**+3.73**	**+1.49**	**+2.24**	**+2.49**	**1.14**

content supplied again in a subsequent call is counted again. Appendix G details the calls included for each method. At each budget, we report the highest validation SR attained within the cumulative expenditure, including the validation calls used to obtain that result. By step 200, cumulative consumption is approximately 550M tokens for CONTINUAL-SKILL and 825M for SLIM. At a total budget of 350M tokens, their best validation SRs are approximately 84% and 54%, respectively.

5.4 ABLATION STUDY: CONTRIBUTIONS OF UTILITY AND SEMANTIC ROUTING

We evaluate three candidate-construction variants on ALFWorld Large using the same three seeds and training configuration. Semantic-Only uses semantic information throughout candidate construction, with utility-based ranking and penalties disabled. Utility-Only uses utility information throughout candidate construction, with semantic filtering, semantic ranking, and embedding-based diversity disabled. These ablations cover both exploitation and coverage selection. Full-CONTINUAL-SKILL combines semantic relevance and utility using the routing rules in Appendix C. All variants retain the same candidate budgets, exposure procedure, policy updates, and skill-management settings. Table 4 reports their test success rates.

Table 4: Ablation of semantic and utility routing on ALFWorld Large ($10\times$) initial bank, using the same 3 seeds and training configuration. Values are mean $\pm$ sample standard deviation.

Configuration	**SR (%)**
Semantic-Only	76.37 ± 1.55
Utility-Only	80.60 ± 0.75
Full **CONTINUAL-SKILL**	$\mathbf{81.34 \pm 0.75}$

The ablation shows that combining semantic and utility scores achieves the highest mean SR on ALFWorld Large, supporting their joint use for candidate selection of CONTINUAL-SKILL.

6 DISCUSSION

Potential extensions to inference-time routing and AutoResearch. With model parameters fixed, CONTINUAL-SKILL can update skill utility from task feedback independently of training, adapting skill selection across policy updates, model replacements, and third-party upgrades. A brief utility warm-start uses randomized skill exposure to collect feedback from the new model, refresh utility estimates, and adjust routing priorities and lifecycle states, enabling reuse of the existing skill bank without parameter access or additional model training. This separation also motivates extensions to AutoResearch, where research procedures serve as skills, task executions provide trajectories, and experimental evaluations supply feedback. As the underlying model and research tasks evolve, utility updates can guide which procedures to reuse, reprioritize, or retire.

Task-domain partitioning and candidate coverage. Task grouping plays a central role in agent post-training by organizing experience retrieval, evaluation, and reuse. In CONTINUAL-SKILL, task groups define skill applicability domains and historical feedback baselines, connecting candidate selection with utility estimation. This structure creates a feedback loop: relevant skills shape training trajectories, and trajectory outcomes guide subsequent skill selection within each domain. Task groups therefore provide a practical interface between external knowledge and policy learning. Thus,

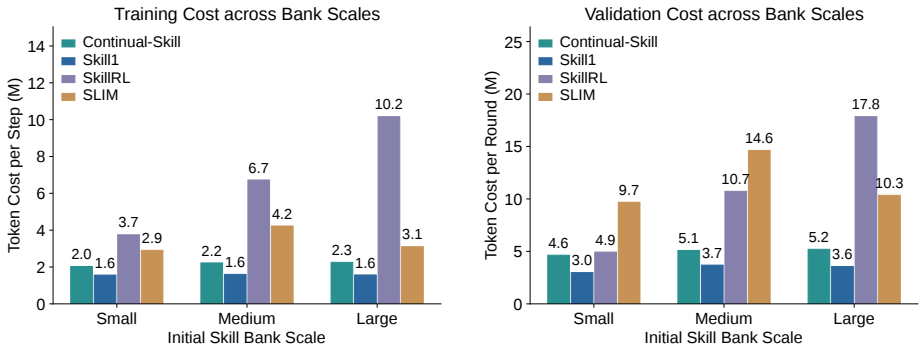


Figure 5: **Token costs on ALFWorld.** Training cost per step and validation cost per round under different skill-bank conditions. The costs of Skill1 and CONTINUAL-SKILL, whose maximum retrieval or exposure budgets remain bounded as the bank grows, do not vary substantially with bank size; SkillRL and SLIM, which manage skill lifecycles, both show some increase in overhead as the number of skills in the bank grows, most noticeably for SkillRL.

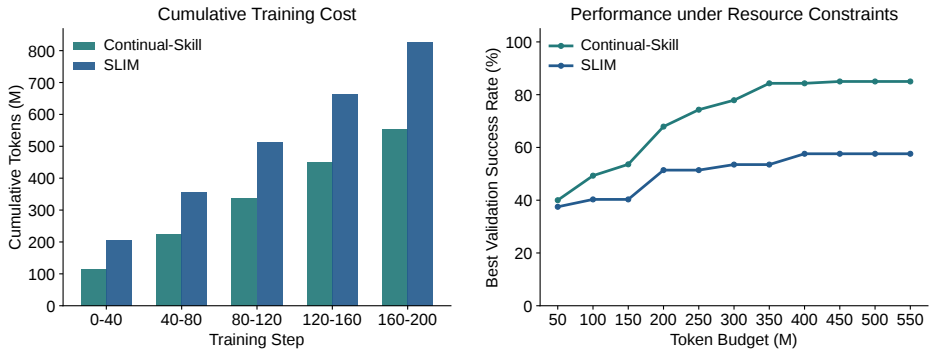


Figure 6: **Resource-indexed comparison with SLIM on Large ALFWorld.** Left: cumulative full-workflow token consumption at successive 40-step endpoints. Right: the highest validation SR achieved within each total token budget.

joint learning to an automatic group partitioning approach can adapt task groups and share procedures across related tasks. Such adaptive partitioning is especially valuable for AI Scientist systems. Scientific agents face broader, longer-horizon, and continuously evolving tasks. Utility-guided domain routing turns this heterogeneity into effective knowledge transfer by sharing procedures supported by evidence, containing cross-domain interference, and continually reorganizing the skill bank around demonstrated scientific value. Future work will apply the framework to self-evolving AI Scientist systems, supporting cross-domain research.

7 CONCLUSION

We presented CONTINUAL-SKILL, a framework that turns ordinary training trajectories into evidence for both policy learning and skill selection. By reassessing skill utility during training, the system guides experience reuse as the policy evolves, without separate per-skill evaluation rollouts. Experiments on ALFWorld and WebShop demonstrate robust task performance under increasing skill-bank interference. These findings support a practical principle for self-improving agents: use ongoing task feedback to keep reusable experience aligned with the agent's current capabilities. This principle motivates extending utility-guided reuse to scientific auto-research agents, where experimental outcomes can guide the selection of research procedures as models and tasks evolve.

AI USE STATEMENT

AI is used to assist and polish articles, and is employed to assist in script implementation assistance and error correction. It is also used in generation of the initial skill bank (Medium and Large) both in ALFWorld and Webshop benchmarks.

REPRODUCIBILITY STATEMENT

Algorithm 1 and Appendices B–E.2 describe utility estimation, skill routing, lifecycle transitions, new-skill trials, and policy optimization. Appendices F–G detail skill-bank construction, training and evaluation procedures, and metric aggregation. Appendix H provides the interaction prompts.

REFERENCES

Agent Skills. Agent Skills specification, 2025. URL https://agentskills.io/specification. Accessed: 2026-09-15.

Google. Gemini 2.5 Flash. Google AI for Developers, 2025. URL https://ai.google.dev/gemini-api/docs/models/gemini-2.5-flash. Accessed: 2026-09-23.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023. URL https://arxiv.org/abs/2308.00352.

Wenxuan Huang, Mingyu Tsoi, Yanhao Huang, Xinjie Mao, Xue Xia, Hao Wu, Jiaqi Wei, Yuejin Yang, Lang Yu, Cheng Tan, Xiang Zhang, Zhangyang Gao, and Siqi Sun. HarmonyCell: Automating single-cell perturbation modeling under semantic and distribution shifts. *arXiv preprint arXiv:2603.01396*, 2026. URL https://arxiv.org/abs/2603.01396.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. AIDE: AI-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*, 2025. URL https://arxiv.org/abs/2502.13138.

Hongyi Liu, Haoyan Yang, Tao Jiang, Bo Tang, Feiyu Xiong, and Zhiyu Li. Skillsvote: Lifecycle governance of agent skills from collection, recommendation to evolution. *arXiv preprint arXiv:2605.18401*, 2026.

Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Skill0: In-context agentic reinforcement learning for skill internalization. *arXiv preprint arXiv:2604.02268*, 2026.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Qwen Team. Qwen3-Embedding-0.6B. Hugging Face model repository, 2025. URL https://huggingface.co/Qwen/Qwen3-Embedding-0.6B. Accessed: 2026-09-20.

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

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

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

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

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.

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

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023. URL https://arxiv.org/abs/2308.08155.

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

Xue Xia, Chengkai Yao, Mingyu Tsoi, Xinjie Mao, Wenxuan Huang, Jiaqi Wei, Hao Wu, Cheng Tan, Lang Yu, Yuejin Yang, Mengdi Liu, Siqi Sun, and Zhangyang Gao. AblateCell: A reproduce-then-ablate agent for virtual cell repositories. *arXiv preprint arXiv:2604.19606*, 2026b. URL https://arxiv.org/abs/2604.19606.

Junlin Yang, Che Jiang, Yu Fu, Tianwei Luo, Can Ren, Weizhi Wang, Kaikai Zhao, Hongyi Liu, Yuxin Zuo, Yuru Wang, Yuchen Fan, Kai Tian, Zhenzhao Yuan, Xiaojian Lin, Li Sheng, Rushi Qiang, Guoli Jia, Xingtai Lv, Ermo Hua, Dianqiao Lei, Youbang Sun, Ning Ding, Bowen Zhou, and Kaiyan Zhang. Frontis-MA1: Training an AI4AI model towards recursive self-improvement in machine learning engineering. *arXiv preprint arXiv:2607.28568*, 2026a. URL https://arxiv.org/abs/2607.28568.

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

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. Autoskill: Experience-driven lifelong learning via skill self-evolution. *arXiv preprint arXiv:2603.01145*, 2026c.

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*, volume 35, 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 *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Barry Zhang, Keith Lazuka, and Mahesh Murag. Equipping agents for the real world with agent skills. Anthropic Engineering, October 2025. URL https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills. Accessed: 2026-09-15.

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

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 62743–62767, 2024. URL https://proceedings.mlr.press/v235/zhuge24a.html.

A Related Work

Language agents and scientific workflows. GPTSwarm represents language agents as computational graphs and optimizes node prompts and graph connectivity; MetaGPT organizes collaborative execution through roles and standard operating procedures; AutoGen organizes tool use and human or model participation through multi-agent conversations. These systems specify how component operations exchange information to execute broader workflows(Zhuge et al., 2024; Hong et al., 2023; Wu et al., 2023). Execution-centered agents also alternate reasoning with actions or retain verbal reflections across attempts, using environmental observations and previous outcomes to shape subsequent behavior(Yao et al., 2023; Shinn et al., 2023). In scientific and machine-learning workflows, AIDE searches a tree of executable code solutions; some scientific agents also organize experiments around executable candidates and observed outcomes, but use different search spaces and verification procedures(Jiang et al., 2025; Huang et al., 2026; Xia et al., 2026b).

Skill internalization and learned self-improvement. External procedural context can provide training-time guidance that is subsequently transferred into policy parameters. SKILL0 progressively withdraws skill context under a curriculum, while Skill-SD conditions a teacher on skills and distills its guidance into a student operating with a plain task prompt (Lu et al., 2026; Wang et al., 2026). Other approaches learn the operations used to acquire and exploit experience: SkillRL distills trajectories into a hierarchical bank and recursively evolves it during reinforcement learning, while Skill1 trains a policy's selection, utilization, and distillation from task-outcome feedback. FrontisMA1 trains Draft, Improve, Debug, and Crossover program-evolution operators through execution-grounded supervised learning and reinforcement learning, and composes them into longer search trajectories. These methods couple learned behavior with mechanisms for reusing or producing procedural knowledge, using different forms of external guidance and execution feedback (Xia et al., 2026a; Shi et al., 2026; Yang et al., 2026a).

Skill construction and lifecycle management. Reusable experience can be maintained as an external artifact. Voyager builds a growing library of executable skills through interaction, and ExpeL extracts natural-language insights from experience for later retrieval (Wang et al., 2023; Zhao et al., 2024). Agent Skills packages procedural instructions with supporting resources, while AutoSkill derives, maintains, and retrieves personalized skills from dialogue and interaction traces without retraining the underlying model (Zhang et al., 2025; Agent Skills, 2025; Yang et al., 2026c). Lifecycle frameworks further specify how evidence changes the available library. SLIM evaluates marginal external contribution through leave-one-skill-out validation and uses that assessment for retention, retirement, and expansion during reinforcement learning. SkillsVote combines structured library search with post-execution skill-linked attribution and evidence-gated updates, including online evolution and reuse of frozen libraries (Shen et al., 2026; Liu et al., 2026). SkillForge also focuses on continuous skill verification and evolution: explicit skill calling enables reinforcement learning to directly optimize invocation decisions, while post-invocation success rates and usage counts identify skills for LLM-based review and revision. It also generates new skills from successful, failed, and contrastive trajectories(Yang et al., 2026b).

B Utility Targets, Identification, and Online Statistics

B.1 Conditional exposure effects

Let $\mathcal{F}_i$ contain the task, policy, skill contents, routing state, candidate set, injection probabilities, and historical baseline fixed before trajectory i. For candidate s, let $I_{i,-s}$ denote the other candidates' exposure vector and ξ_i the policy and environment randomness. The potential feedback $R_i(a, I_{i,-s}; \xi_i)$ is the feedback that would result with the exposure of s set to a. The target is

$$\Delta_{i,s} = \mathbb{E}_{I_{i,-s},\xi_i}[R_i(1, I_{i,-s}; \xi_i) - R_i(0, I_{i,-s}; \xi_i) \mid \mathcal{F}_i], \quad s \in C_i. \tag{9}$$

This target averages over the other skill combinations generated by the current router. It accommodates substitution between similar skills and complementarity between cooperating skills. Its value depends on the current policy, task, candidate contents, and assignment mechanism.

Proposition 1 (Conditional unbiasedness under ordinary randomized exposure) *Given $\mathcal{F}_i$, suppose skill s is exposed with its true probability $p_{i,s} \in (0, 1)$, independently of the other skills'*

assignments and trajectory randomness. Suppose observed feedback is consistent with the potential feedback under the realized assignment and has finite expectation. Then the contribution in Eq. (6) satisfies $\mathbb{E}[Z_{i,s} \mid \mathcal{F}_i] = \Delta_{i,s}$.

Proof. Conditioning on the two exposure arms gives

$$\begin{aligned}\mathbb{E}[Z_{i,s} \mid \mathcal{F}_i] &= \mathbb{E}[H_i \mid I_{i,s} = 1, \mathcal{F}_i] - \mathbb{E}[H_i \mid I_{i,s} = 0, \mathcal{F}_i] \\ &= \mathbb{E}[R_i(1, I_{i,-s}; \xi_i) - b_{g_i,t} \mid \mathcal{F}_i] \\ &\quad - \mathbb{E}[R_i(0, I_{i,-s}; \xi_i) - b_{g_i,t} \mid \mathcal{F}_i] \\ &= \Delta_{i,s}.\end{aligned} \tag{10}$$

The first equality uses the true assignment probability. Independence and consistency justify the second. The pre-specified baseline cancels between the two arms. □

Random assignment supplies identification, while the historical baseline centers feedback. The historical baseline centers feedback to reduce task-difficulty variance, while randomized exposure identifies the skill effect under the remaining task heterogeneity. When the second moments exist,

$$\mathbb{E}[Z_{i,s}^2 \mid \mathcal{F}_i] = \frac{\mathbb{E}[H_i^2 \mid I_{i,s} = 1, \mathcal{F}_i]}{p_{i,s}} + \frac{\mathbb{E}[H_i^2 \mid I_{i,s} = 0, \mathcal{F}_i]}{1 - p_{i,s}}. \tag{11}$$

Probabilities near either endpoint can amplify variability. Retaining two arms can provide the support needed for comparison.

B.2 Scope under adaptive retrieval and task grouping

Retrieval changes with historical utility and the policy changes between batches, so $\Delta_{i,s}$ need not be constant over time. A window mean summarizes the local evidence observed when the skill was a candidate. Under a fixed window and a pre-specified assignment mechanism, the contribution in Eq. (6) is conditionally unbiased for the average exposure effect. Under adaptive selection, rolling windows summarize recent candidate contributions. Lifecycle decisions use empirical standard-error thresholds and require minimum observation counts overall and in each exposure arm.

Task groups define applicability domains and historical feedback baselines. Task-specific skills accumulate evidence within their domains; each general skill pools evidence across visited task groups after group-specific centering. Provenance and perturbation labels support traceability and stratified analysis. The estimator uses candidate identities, assignment indicators, probabilities, and feedback.

B.3 Feedback configuration, historical baselines, and online moments

For skill utility estimation on ALFWorld and WebShop benchmarks, we set $(w_y, w_f, w_\ell, w_v) = (1.0, 0.25, 0.1, 0.1)$ in Eq. (4), giving

$$R_i = y_i - 0.25(1 - y_i) - 0.1\frac{L_i}{L_i^{\max}} - 0.1(1 - v_i).$$

The four terms account for task success, task failure, normalized interaction length, and the invalid-action fraction, respectively. We use R_i for skill attribution and historical baseline estimation. GRPO policy advantages use the environment reward r_i^{env} (Appendix E.2).

The historical baseline is

$$b_{g,t} = \frac{1}{N_{g,<t}} \sum_{j<t} \sum_{i \in \mathcal{T}_j : g_i = g} R_i, \tag{12}$$

where $\mathcal{T}_j$ is the completed trajectory set for batch j and $N_{g,<t}$ is the corresponding historical count. A task group with zero historical observations uses $b_{g,t} = 0$. All trajectories in a batch use the same pre-batch baseline. After centering, their feedback is added to the historical statistics for the next batch.

For a new contribution z, the count n, mean u, and centered sum of squares M_2 are updated as

$$\begin{aligned} n' &= n+1, & \delta &= z-u, \\ u' &= u+\delta/n', & M_2' &= M_2+\delta(z-u'), \\ \widehat{V}' &= M_2'/(n'-1), & \widehat{\mathrm{SE}}' &= \sqrt{\widehat{V}'/n'}. \end{aligned} \tag{13}$$

Variance is computed only with at least two observations. Otherwise initialized values are retained, and governance eligibility thresholds prevent decisions based on insufficient evidence. Every candidate opportunity increments either n_s^+ or n_s^-.

Three time scales are maintained: recent moments for routing, cumulative moments for soft penalties, and bounded timestamped contributions for lifecycle audits. When a recent window rolls over, its utility estimate is saved as a fallback and its moment accumulator is cleared. Skills without new observations use the saved routing estimate. This reduces dependence of current ranking on early-policy evidence without treating a lack of new evidence as zero utility.

B.4 Alternative assignment and baseline variants

Fixed-size sampling without replacement can have exactly known marginal probabilities, but including a skill changes the distribution of its companions. The inverse-probability contrast then compares two conditional assignment mechanisms and can include replacement effects. Under deterministic top-ranked injection, each candidate's conditional exposure probability is either zero or one. Proposition 1 requires positive probabilities for both exposure and withholding.

An implementation variant uses a within-task, within-batch leave-one-out baseline,

$$b_{i,t}^{\mathrm{LOO}} = \frac{1}{K_x-1} \sum_{j\in\mathcal{T}_t(x_i),\, j\neq i} R_j, \qquad K_x = |\mathcal{T}_t(x_i)| > 1. \tag{14}$$

The main algorithm uses a historical task-group baseline computed from completed batches.

Table 5: Candidate retrieval rules.

Rule	Retrieval implementation
Scaling and penalty	Within domain d, set $\widetilde{U}_{s,t} = \mathrm{clip}((\widehat{U}_{s,t}^{\mathrm{route}} - m_{d,t})/(a_{d,t}+\varepsilon), -c, c)$. Use the reference-set median $m_{d,t}$ and scale-corrected median absolute deviation $a_{d,t}$ (standard-deviation fallback), preferring currently observed skills. Too few estimates or both scales degenerate yield zero; unobserved skills read saved routing estimates. Set $P_{s,t} = \lambda_P Q_{s,t} \exp(-A_{s,t}/\tau_P)$, where $A_{s,t}$ counts audit periods since new evidence and $Q_{s,t}$ indicates $n_s \geq n_0^{\mathrm{r}}$, $\min(n_s^+, n_s^-) \geq m_0^{\mathrm{r}}$, and $\bar{U}_s^{\mathrm{cum}} + z_{\mathrm{r}}\mathrm{SE}_s^{\mathrm{cum}} \leq \delta_{\mathrm{r}}$. This comparison serves as the decision threshold in the routing penalty, with evidence gates preventing decisions under insufficient evidence.
Ranking and selection	Exploitation scores are $B_{i,s}^{\mathrm{use}} = \beta_E E_{i,s} + \beta_U \widetilde{U}_{s,t} - P_{s,t}$; coverage scores are $B_{i,s}^{\mathrm{cov}} = \beta_E E_{i,s} + \alpha \min(1, A_{s,t}/\tau_A)$. Each channel forms a high-scoring preliminary pool, then greedily maximizes $B_{i,s} - \lambda_D \max(0, \max_{u\in C} e_s^\top e_u)$, with zero similarity penalty for an empty selection. Exploitation selects up to K_d^{use} entries; coverage selects up to K_d^{cov} additional candidates from the remaining entries. Within this budget, cold-start slots are allocated to eligible new skills as described in Appendix E. Remaining coverage slots use the coverage score and diversity rule above to select semantically relevant skills with stale evidence.

C Utility-Guided Candidate Retrieval and Exposure

Candidate construction and retrieval implementation. The task text and initial observation query the general and corresponding task domains. We use Qwen3-Embedding-0.6B (Qwen Team, 2025) to encode task queries and skills, keeping its parameters frozen throughout training. Relevance is $E_{i,s} = e_i^\top e_s$ for normalized embeddings. Table 5 specifies the Full configuration.

The routing ablations in Table 4 apply to the complete candidate-construction procedure. Semantic-Only retains semantic filtering, ranking, and diversity selection while disabling utility-derived scores

and penalties. Utility-Only constructs candidates from utility evidence, with semantic filtering, semantic scores, and embedding-based diversity disabled. Each variant applies its selected information source across both exploitation and coverage selection, under the same candidate budgets.

Routing configuration. The routing parameters used on ALFWorld and WebShop are listed in Table 6. Evidence age $A_{s,t}$ counts audit periods since the skill last received new evidence. Audits occur every five training steps, so $\tau_A = \tau_P = 2$ corresponds to ten training steps.

Table 6: Routing parameters for CONTINUAL-SKILL on ALFWorld and WebShop.

Parameter or setting	Value
Semantic relevance weight β_E	1.0
Normalized utility weight β_U	0.5
Coverage amplitude α	0.3
Coverage time scale τ_A	2 audit periods
Diversity penalty λ_D	0.3
Negative-utility penalty amplitude λ_P	1.0
Penalty decay time scale τ_P	2 audit periods
Soft-penalty standard-error coefficient z_{r}	1.0
Soft-penalty upper-bound cutoff δ_{r}	0.0
Minimum cumulative count n_0^{r}	40
Minimum cumulative count per exposure arm m_0^{r}	12
Normalized utility clipping amplitude c	3.0
Numerical stabilizer ε	10^{-8}
Preliminary candidate limit per domain	30
Exploitation candidates per domain K_d^{use}	8
Coverage candidates per domain K_d^{cov}	2

With these values, the routing scores and soft penalty are

$$\begin{aligned} B_{i,s}^{\mathrm{use}} &= E_{i,s} + 0.5\widetilde{U}_{s,t} - P_{s,t}, \\ B_{i,s}^{\mathrm{cov}} &= E_{i,s} + 0.3\min(1, A_{s,t}/2), \\ P_{s,t} &= Q_{s,t}\exp(-A_{s,t}/2). \end{aligned}$$

The indicator $Q_{s,t}$ uses the cumulative moments and exposure-arm counts defined above.

Ordinary exposure and logging. Let $M_{i,d} = |C_i^d|$ be the final domain candidate count. With expected injection budget $\kappa_d > 0$, set

$$p_{i,s} = \begin{cases} 1/2, & M_{i,d} = 1, \\ \min(\kappa_d, M_{i,d} - 1)/M_{i,d}, & M_{i,d} > 1, \end{cases} \qquad s \in C_i^d. \tag{15}$$

We use $\kappa_d = 2$ for the general-skill domain and $\kappa_d = 6$ for the task-specific domain. With ten candidates in each domain, the exposure probabilities are 0.2 and 0.6, respectively, giving expected injection counts of two general and six task-specific skills. For smaller candidate sets, the rule above adapts the probabilities to $M_{i,d}$.

Empty domains supply no candidates. Conditional on pre-interaction information, ordinary exposures are independent Bernoulli draws with these probabilities and are independent of trajectory randomness. Before interaction, we log the actual probabilities, final candidate identities, realized assignments, and policy/routing versions. Each candidate contributes one observation to its realized exposure arm. Both arms have positive probability, and the realized injection count varies across trajectories. New skills receive reserved coverage candidate slots for five training steps, beginning at the next step after registration. Each domain provides up to two coverage slots, with random selection when eligible new skills exceed the available slots. Their utility contributions are computed from the recorded assignments, exposure probabilities, and trajectory feedback using Eq. (6). Appendix E describes the cold-start procedure.

D EVIDENCE-BASED SKILL LIFECYCLES

States and audit criteria. The states are exploration-enabled E, mature active A, probation P, suspended S, and archived X. States E and A participate in ordinary service; P and S can receive training audit candidates; X leaves retrieval while retaining its records. At audit time t, view $h \in \{\text{safe}, \text{worth}\}$ uses at most M_h candidate contributions from the last W_h steps. Their mean, empirical standard error, total count, and arm counts define

$$\begin{aligned} \ell_s^h &= \bar{Z}_s^h - z_h \widehat{\mathrm{SE}}_s^h, \\ u_s^h &= \bar{Z}_s^h + z_h \widehat{\mathrm{SE}}_s^h, \\ G_s^h &= \mathbf{1}\{n_s^h \geq n_0,\ \min(n_s^{h,+}, n_s^{h,-}) \geq m_0\}, \\ D_s &= G_s^{\text{safe}} \mathbf{1}\{u_s^{\text{safe}} < \delta_-\}, \\ B_s &= G_s^{\text{safe}} \mathbf{1}\{\ell_s^{\text{safe}} > \delta_+\}, \\ \delta_- &< 0 < \delta_+. \end{aligned} \tag{16}$$

The evidence gate G_s^h requires sufficient total observations and observations in both exposure arms. The bounds ℓ_s^h, u_s^h and cutoffs $\delta_-, \delta_+, \delta_{\text{worth}}$ define the heuristic lifecycle decision rules. Consecutive audit counters c_s^- and c_s^+ count satisfaction of D_s and B_s, respectively, and reset when their criterion fails. The required confirmation counts are q_- and q_+. Table 8 applies the first matching rule to each skill's own evidence.

Audit configuration. Lifecycle audits and recent routing-window refreshes occur every five training steps on both benchmarks. Each audit view uses the preceding 60 training steps, subject to its per-skill observation capacity. Table 7 lists the audit parameters.

Table 7: Lifecycle audit parameters for CONTINUAL-SKILL on ALFWorld and WebShop.

Parameter or setting	Value
Lifecycle audit interval	5 training steps
Recent routing-window refresh interval	5 training steps
Safety window W_{safe}	60 training steps
Safety capacity M_{safe}	320 observations per skill
Benefit window W_{worth}	60 training steps
Benefit capacity M_{worth}	1600 observations per skill
Minimum total count n_0	40
Minimum count per exposure arm m_0	12
Standard-error coefficients $z_{\text{safe}}, z_{\text{worth}}$	Both 2.8652602385321333
Negative-utility cutoff δ_-	-0.05
Positive-utility cutoff δ_+	$+0.05$
Minimum-benefit cutoff δ_{worth}	0.10
Negative confirmation count q_-	2 consecutive audits
Positive confirmation count q_+	2 consecutive audits

Table 8: Lifecycle transition rules, applied in order.

Priority	Evidence and current state	Update
1	Current state is X.	Remain archived.
2	$D_s = 1$.	Suspend while negative confirmations are below q_-; archive once the threshold is reached.
3	Current state is S and $c_s^+ \geq q_+$.	Recover to P and restart promotion confirmation.
4	Current state is E or P, and $c_s^+ \geq q_+$.	Promote to A.
5	Current state is E, P, or S; $G_s^{\text{worth}} = 1$, $\bar{Z}_s^{\text{worth}} \geq \delta_-$, and $u_s^{\text{worth}} < \delta_{\text{worth}}$.	Archive for insufficient minimum benefit.
6	Otherwise.	Keep the state and await further evidence.

E AUXILIARY SKILL GENERATION AND TRIALS

E.1 GENERATION CONFIGURATION

The generation module uses Gemini 2.5 Flash (Google, 2025) to propose a bounded number of new skills from recent trajectories after persistent task-domain failures. Proposals undergo structure and duplication checks before registration with new identities and fresh utility statistics. Table 9 specifies the generation and screening rules, and Table 10 lists the generation and cold-start parameters used on ALFWorld and WebShop. Appendix H provides the generation prompt.

Cold-start scheduling. Each domain provides up to two coverage/exploration candidate slots. During cold start, slots within this budget are reserved for newly registered skills. When eligible new skills exceed the available slots, the router randomly samples a subset to fill them. A skill registered after training step t is eligible during steps $t+1$ through $t+5$. These reservations provide early candidate opportunities. Recorded exposure assignments, probabilities, and trajectory feedback enter the utility estimator in Eq. (6), and the resulting statistics guide subsequent routing and lifecycle decisions.

Table 9: Skill generation rules.

Item	Implemented rule
Trigger	Smooth domain batch failure rate: $e_{d,t} = \alpha_f f_{d,t} + (1-\alpha_f)e_{d,t-1}$. Set the failure mark after $q_f = 3$ consecutive training batches with $e_{d,t} > \eta_{\text{in}} = 0.30$; clear it after three consecutive training batches with $e_{d,t} < \eta_{\text{out}} = 0.24$.
Proposals	Propose at most $B_{\text{new}} = 3$ entries per triggered domain per generation cycle. Accepted entries receive new identities and fresh statistics; existing contents are not overwritten. "Rewrite" in Figure 2 denotes proposal construction.
Screening	Validate structure; reject duplicates of same-domain pre-generation catalog entries (including archived ones): $e_s^\top e_u \geq \rho_{\text{dup}}$ and $\psi(s) = \psi(u)$ for normalized behavioral signature ψ.
Locks	All new skills archived marks an unsuccessful batch. Repeated all-duplicate proposals or excessive unsuccessful batches lock generation. Implementation-defined context-fingerprint changes can reset controls. Pending proposals and empty outputs follow scheduler state.

Table 10: Generation and cold-start parameters for CONTINUAL-SKILL on ALFWorld and WebShop.

Parameter	Value	Description
α_f	0.25	EWMA coefficient for task-domain failure rates.
q_f	3	Consecutive training batches required to set or clear the failure mark.
η_{in}	0.30	Failure-mark entry threshold.
η_{out}	0.24	Failure-mark exit threshold.
B_{new}	3	Maximum skill proposals per triggered domain per generation cycle.
ρ_{dup}	0.92	Cosine-similarity threshold used jointly with matching behavioral signatures for duplicate screening.
Coverage slots per domain	At most 2	Candidate slots used for cold-start reservations.
Cold-start duration	5 steps	Begins at the next training step after registration.
Oversubscribed slots	Random sampling	Select a subset of eligible new skills to fill the available slots.

E.2 POLICY UPDATES FROM ENVIRONMENT REWARD

For repeated attempts of task x, GRPO advantages are

$$\widehat{A}_{x,k} = \frac{r^{\text{env}}_{x,k} - \bar{r}^{\text{env}}_{x}}{\sigma^{\text{env}}_{x} + \varepsilon}. \tag{17}$$

Let $\rho_{i,\ell}(\theta)$ denote the ratio of current-policy to sampling-policy probabilities at an action-generation position. The actor objective is

$$\begin{aligned}\mathcal{L}_{\text{actor}}(\theta) = & -\mathbb{E}_{i,\ell}\Big[\min\Big(\rho_{i,\ell}(\theta)\widehat{A}_i, \text{clip}(\rho_{i,\ell}(\theta), 1-\epsilon_c, 1+\epsilon_c)\widehat{A}_i\Big)\Big] \\ & + \beta_{\text{KL}}\mathcal{R}_{\text{KL}}(\theta).\end{aligned} \tag{18}$$

We use $\epsilon_c = 0.2$ and $\beta_{\text{KL}} = 0.01$ on ALFWorld and WebShop. The clipping interval is $[0.8, 1.2]$, and β_{KL} weights the KL regularization term. Ratios and masks are evaluated over valid generated action tokens. Task groups $g(x)$ define skill domains and historical feedback baselines; repeated attempts of the same task define the relative policy advantages.

E.3 Batch boundaries and evidence identities

Candidate sets and exposure are fixed before environment interaction. After all trajectories finish, contributions are computed from pre-batch baselines before group means are updated. Routing utility is then normalized, periodic audits change lifecycle states and register proposals, and windows roll according to their schedule. The policy consumes the same recorded batch. Both updates affect subsequent batches.

Candidate identities, actual assignments, probabilities, policy versions, routing versions, and skill records associate feedback with the knowledge context that produced it. Archival preserves records and new entries receive independent identities, avoiding accidental mixing of different contents under one utility statistic. Training permits audit candidates in prompts; ordinary evaluation retrieves service-enabled states. Our experiments evaluate utility adaptation during policy training, where every feedback update is tied to a recorded policy, routing state, and skill assignment. The same update interface defines a direct extension to inference-time adaptation under a dedicated online-evaluation protocol.

E.4 Scope of the computational claim

Utility estimation processes the N training trajectories with $O(NK_C)$ bookkeeping. The retrieval index caches within-domain pairwise skill similarities, requiring $O(\sum_d B_d^2)$ space for B_d retrievable skills in domain d. Embedding retrieval and diversity selection contribute additional computation as banks grow.

Figure 5 non-overlapping mean token costs for the training phase and periodic validation. Figure 6 accumulates tokens across training interaction, method-specific auxiliary calls, and periodic validation. Calls shared across operations are counted once. Each validation result is associated with the cumulative expenditure through completion of that validation round.

F Skill-Bank Construction and Quality Checks

F.1 Nested construction protocol

We normalize the initial skill bank from SkillRL (Xia et al., 2026a) to obtain the base bank $\mathcal{B}_0$ of size N, which forms the Small bank. Using a common construction seed, we generate $9N$ additional entries. Fixed prefixes define the expanded banks:

$$\begin{aligned}\mathcal{B}^{\text{Small}} &= \mathcal{B}_0, \\ \mathcal{B}^{\text{Medium}} &= \mathcal{B}_0 \cup \{s_j\}_{j=1}^{4N}, \\ \mathcal{B}^{\text{Large}} &= \mathcal{B}_0 \cup \{s_j\}_{j=1}^{9N}.\end{aligned} \tag{19}$$

The base preserves general and task-specific domains and is converted to each method's input format. Every expansion retains a stable identity, parent skill, domain, and perturbation type. The common seed and content identities provide a way to check equivalent inputs across methods.

The sequence is stably sorted by category and other fields before prefix selection. Medium therefore concentrates on conflicts, mismatches, overspecialization, and some incomplete entries; Large adds more incomplete entries and many paraphrases. Table 12 reports the construction counts.

Table 11: Perturbation families used to construct expanded banks.

Family	Content transformation	Intended stressor
Redundant paraphrase	Reword the core procedure.	Duplicate retrieval competition and context occupancy.
Overspecialization	Add narrow conditions or local steps.	Misapplied guidance outside its preconditions.
Incomplete or outdated	Omit a necessary operation or retain an obsolete assumption.	Plausible but incomplete procedural guidance.
Cross-task mismatch	Assign experience to an unsuitable task context.	Irrelevant skills within a domain.
Plausible conflict	Preserve task vocabulary while inserting incompatible advice.	Divergence between semantic relevance and actual usefulness.

Table 12: Counts of added entries, column totals equal $4N$ or $9N$.

	ALFWorld		WebShop	
Perturbation	$5\times$	$10\times$	$5\times$	$10\times$
Redundant paraphrase	0	159	0	195
Overspecialization	80	80	98	98
Incomplete or outdated	18	79	22	97
Cross-task mismatch	39	39	48	48
Plausible conflict	39	39	48	48
Total added	176	396	216	486

The categories identify the construction procedures used to expand the skill banks. Each entry's effect depends on the task and current policy.

G Experimental Protocol and Reporting Scope

G.1 Shared training and evaluation settings

Skill1, SkillRL, SLIM, CONTINUAL-SKILL, and both governance-ablation variants use the training configuration in Table 13. Each step contains eight tasks with eight rollouts per task, yielding a shared budget of 64 base training trajectories.

Token accounting sums input and output tokens for training interaction, skill-related auxiliary calls, and periodic validation. For Skill1, auxiliary calls include query generation, skill reranking, and skill distillation when enabled, primarily performed by the training model. For SkillRL, they include third-party API calls for failure-trajectory analysis and skill generation or updates. For SLIM, they include additional MEC/skill-audit rollouts and third-party API calls for skill generation. For Full CONTINUAL-SKILL, they include third-party API calls for skill generation; new-skill trials use ordinary training rollouts and are counted under training interaction. Utility statistics and local lifecycle decisions are computed locally and contribute no LLM tokens.

Each LLM call contributes its input and output tokens once to the cumulative total, including repeated prompt content across calls. The total aggregates tokens from the training model and third-party models, together with periodic validation calls for all four methods. Figure 6 reports performance as a function of this cumulative total.

Method-specific retrieval and skill injection. We evaluate benchmark-adapted implementations of Skill1, SkillRL, and SLIM (Shi et al., 2026; Xia et al., 2026a; Shen et al., 2026) on common SkillRL-derived banks, interaction prompts, benchmark interfaces, and training settings. Retrieval, selection, and skill-injection counts follow the method-specific configurations in Table 14.

For Full CONTINUAL-SKILL, we distinguish between the candidate-set size and the realized number of skills injected into the prompt. For each task attempt, the router constructs candidates separately from the general-skill domain and the task-specific domain associated with the current task. In each

Table 13: Shared training configuration for all four compared methods and the governance ablation. Resource allocation is per task.

Item	Configuration
Initial model	Qwen2.5-7B-Instruct
Benchmark	ALFWorld and WebShop
Training steps	200
Tasks per step	8
Rollouts per task	8
Base training trajectories per step	64; excludes additional audit, distillation, and evaluation trajectories
Learning rate	10^{-6}
Training sampling temperature	1.0
Validation sampling temperature	0.4; sampling enabled
Validation frequency	Every 10 training steps
Checkpoint selection metric	Validation Success Rate on both benchmarks
Maximum prompt length	8,192 tokens
Interaction limit per trajectory	ALFWorld: 30; WebShop: 15
Resources per task	4 NVIDIA H200-equivalent GPUs, 80 CPU cores, 800 GB RAM
Training parallelism	TP1 $\times$ 4 (tensor parallelism degree 1)
Training time	Approximately 20 hours for 200 steps under TP1 $\times$ 4

Table 14: Method-specific retrieval and skill-injection configurations. Candidate limits and realized prompt-injection counts are reported separately. For Full CONTINUAL-SKILL, "$8+2$" denotes up to eight exploitation candidates and two exploration (coverage) candidates per domain. The reported injection counts are expectations under Bernoulli exposure, not deterministic quotas.

Method	General skills	Task-specific skills	Retrieval and injection
Skill1	No separate quota	No separate quota	Mixed retrieval returns at most five candidates. During training, the model reranks the candidates and retains the top-ranked skill; evaluation retains the candidate list.
SkillRL	12 (match the original general skill number)	6 by default	Skills are matched by task type and inserted through the given prompt template.
SLIM (ALFWorld)	At most 12	At most 3	Skills are ranked by embedding similarity; task-specific skills must also pass the similarity threshold.
SLIM (WebShop)	At most 15	At most 3	Skills are ranked by embedding similarity; task-specific skills must also pass the similarity threshold.
CONTINUAL-SKILL	At most 10 (candidates)	At most 10 (candidates)	Candidates are independently exposed through Bernoulli sampling. With 10 candidates in each domain, the per-candidate exposure probabilities are 0.2 and 0.6, yielding an expected total of 8 injected skills and an expected general-to-task-specific ratio of $1:3$.

domain, the final candidate set contains at most ten skills: up to eight selected by the exploitation channel and up to two selected by the exploration (coverage) channel.

Each candidate is independently exposed through Bernoulli sampling. With ten candidates in each domain, exposure probabilities of 0.2 for general skills and 0.6 for task-specific skills yield an expected total of eight injected skills in a $1:3$ ratio. The realized count varies across trajectories. Table 14 summarizes the method-specific configurations.

Prompt-template alignment. Within each benchmark, all four methods share the interaction prompt template, including system instructions, task and observation formatting, interaction history, action-output requirements, and the skill-context insertion point. Each method supplies its selected skill content through this shared interface using the retrieval and injection procedures in Table 14. Appendix H reproduces the interaction templates used in the reported runs.

Checkpoint selection and reporting for Table 2. We evaluate checkpoints every 10 training steps on the validation splits provided by ALFWorld and WebShop, using Success Rate within each benchmark. The validation and final-evaluation task sets are disjoint. At each reported training step, we select the checkpoint with the highest validation Success Rate among those evaluated up to that step. Table 2 reports the selected checkpoints' final test performance under a fixed matched seed.

For final evaluation, ALFWorld uses the complete `valid_unseen` split of 134 tasks. WebShop uses the small 1,000-product catalog and the complete test split of 500 tasks.

Each training run contains 200 steps with 64 base trajectories per step, yielding 12,800 base training trajectories, together with validation and method-specific auxiliary computation. A complete run takes approximately 20 hours under the hardware configuration in Table 13.

Figure 4 reports training and validation dynamics across three independent seeds (372, 373, and 374), with 95% confidence intervals for validation performance. Tables 3 and 4 report three-seed component analyses using the same training configuration. In the paired governance experiment, No-management retains utility recording and routing while disabling lifecycle changes and skill generation. These paired runs form a separate component experiment from the runs reported in Table 2.

G.2 Metrics and aggregation

For evaluation records $\mathcal{E}$ after removing batch-padding duplicates, let y_i be success and $q_i \in [0, 1]$ the WebShop requirement-matching score. Under equal record weighting,

$$\mathrm{SR} = \frac{100}{|\mathcal{E}|} \sum_{i \in \mathcal{E}} y_i, \qquad \mathrm{Score} = \frac{100}{|\mathcal{E}|} \sum_{i \in \mathcal{E}} q_i. \tag{20}$$

WebShop Score comes from the task-matching field rather than the shaped training reward. If multiple records refer to the same shopping goal, evaluation must consistently specify either equal record weighting or aggregation within goals first.

Overall ALFWorld SR is task-count weighted:

$$\mathrm{SR}_{\mathrm{all}} = \sum_{g \in \mathcal{G}} \frac{N_g}{\sum_h N_h} \mathrm{SR}_g. \tag{21}$$

Here N_g is the number of evaluation tasks in category g, so each task receives equal weight in the overall SR. The arithmetic mean of the six category SRs instead assigns equal weight to categories.

H PROMPT TEMPLATES

ALFWorld Interaction Prompt

```
You are an expert agent operating in the ALFRED Embodied Environment.
Your task is to:
{task_description}
{skills}
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 about the current situation.
This reasoning process MUST be enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose an admissible
action for current step and present it within <action> </action> tags.
```

WebShop Interaction Prompt

```
You are an expert autonomous agent operating in the WebShop
e-commerce environment.
{skills}
Skills are suggestions and may conflict. Follow the current task,
observation, and admissible actions first. Prefer directly applicable
task-specific advice over general advice. Ignore conflicting,
irrelevant, stale, or unsupported advice, and never combine
contradictory instructions.
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 at action {current_step} of at most {max_steps};
{remaining_steps} action(s) remain, including this action.
Your current observation is: {current_observation}.
Your admissible actions of the current situation are:
[
{available_actions}
].

Complete the purchase with this short workflow:
1. Search using the product type and at most two distinctive
functional attributes.
2. Open the most plausible product from the first results page. Avoid
repeating a query.
3. On the product page, select every required variant such as color
or size.
4. Check the required properties and price, then click Buy Now
immediately when they match. Do not leave a matching configurable
product before selecting its variants.

Reply with at most three short lines inside <think> </think>: current
page/progress, unmet constraints, and the next action. Then output
exactly one listed admissible action inside <action> </action>.
```

New-Skill Generation Prompt

```
You manage one domain of an online agent skill bank. Generate concise
actionable skill text.
Return exactly one JSON object with arrays `additions` and
`rewrites`.
Return exactly `addition_budget` valid items in `additions` and
exactly `rewrite_budget` valid items in `rewrites`.
When `addition_budget` is positive, `additions` must never be empty.
Every item needs `title`, `when_to_apply`, and `principle`.
Every rewrite also needs the unchanged `parent_skill_id` supplied in
the request.
Do not decide whether to archive or retain skills. Do not include
prose outside JSON.

REQUEST:
{request_json}
```