LEARNING TO ACCUMULATE KNOWLEDGE WITH MUTUAL INFORMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) agents can improve their performance by reusing knowledge distilled from past interactions. However, curating new experiences into a knowledge bank that becomes more useful as it grows remains challenging. Effective knowledge accumulation should limit redundant overlap among entries and ensure that new knowledge contributes beyond what the bank already provides. Yet training a curator with Group Relative Policy Optimization (GRPO) on standalone task success can reinforce general guidance even when it duplicates existing knowledge. Therefore, we propose **Knowledge Weaver**, a reinforcement learning framework that trains a language model to curate reusable knowledge from agent trajectories. We couple feedback inspired by token-wise mutual information (MI) with marginal success rewards to guide knowledge accumulation. Together, these signals encourage the curator to preserve distinct information from experience and produce entries that improve task success when added to existing knowledge. Standalone success rewards also favor entries that are useful on their own. On ALFWorld and WebShop, Knowledge Weaver achieves mean success rates of 54.0% and 42.0% with $k = 10$ retrieved entries, exceeding GRPO by 16.9 and 18.7 percentage points, respectively. Its knowledge banks also outperform the evaluated prompt-based and established banks, including human-written banks, in overall ALFWorld success rate and WebShop score with the executor frozen. Our codebase is available at this url.

1 INTRODUCTION

Large language model (LLM) agents interact with external environments to solve complex tasks (Yao et al., 2022b; Gur et al., 2024; Yang et al., 2024). Reusing knowledge from these interactions offers a route to agent self-improvement, allowing experience from previous tasks to inform future decisions. Prior work shows that high-quality, task-specific knowledge can improve agent performance (Wang et al., 2024; Li et al., 2026; Ouyang et al., 2026). Such knowledge includes reusable workflows and executable skills that guide action sequences (Wang et al., 2023; 2024; Yuan et al., 2024), as well as distilled lessons about environmental constraints and effective strategies (Majumder et al., 2023; Chen et al., 2024). Stored in external knowledge banks, relevant entries can be retrieved and incorporated into the agent's context without updating the executor's parameters (Zheng et al., 2024; Fu et al., 2024). This makes knowledge accumulation a means of retaining experience beyond individual interactions and supporting its reuse across tasks.

While these benefits establish the value of using knowledge, producing it reliably remains a separate challenge. Existing work relies on pipeline design or training-based methods for knowledge curation. Prompt-based methods extract reflective lessons, conditional guidelines, and reusable workflows from trajectories (Shinn et al., 2023; Zhao et al., 2024; Fu et al., 2024; Wang et al., 2024), then consolidate them through memory or playbook revision (Zhang et al., 2026; Ouyang et al., 2026; Suzgun et al., 2026). Training-based methods integrate knowledge into policy improvement through skill internalization (Lu et al., 2026) or the joint evolution of task execution and reusable skills (Xia et al., 2026; Shi et al., 2026; Yao et al., 2026). When curation is rewarded through downstream task success, this feedback assesses whether generated knowledge helps execution. However, these advances in knowledge construction and use leave a distinct question of how to train a curator to produce better knowledge for a fixed executor. Addressing this question requires considering both the utility of individual entries and what they contribute as knowledge accumulates.

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.

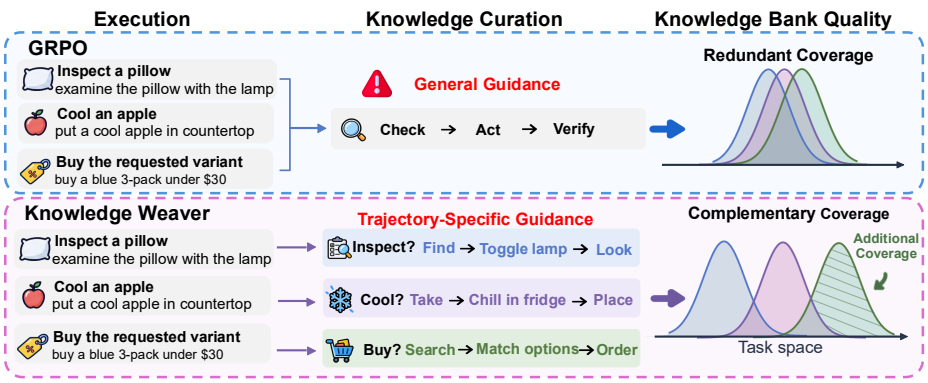


Figure 1: Conceptual illustration of general guidance and complementary knowledge. General guidance can remain useful for individual tasks while providing limited additional information across entries. Knowledge Weaver combines MI-inspired feedback with execution rewards to encourage experience-dependent knowledge and incremental utility.

Building on this, we propose **Knowledge Weaver**, an RL framework that trains a language model as a knowledge curator to extract reusable knowledge from agent trajectories. A good curator should produce knowledge that improves task performance and adds useful information as the bank grows. Effective accumulation therefore calls for both limited redundant overlap and substantial incremental utility. Figure 1 illustrates these considerations through knowledge coverage over a task space. Optimizing only for standalone task success can favor general guidance that repeatedly covers the same tasks, yielding substantial overlap and limited additional coverage as the bank grows. Complementary entries extend the bank's coverage, with the shaded region indicating the incremental contribution of a new entry. We therefore approach knowledge accumulation through both the distinct information retained across entries and the additional utility each entry provides.

Knowledge Weaver couples token-level feedback inspired by mutual information (MI) with marginal success rewards to preserve distinct information and encourage incremental contributions to the bank. The MI-inspired signal contrasts generation probabilities across trajectory contexts, assigning greater credit to context-sensitive tokens than to tokens whose probabilities remain similar across experiences. To reward incremental contributions, we measure the gain in task success when a new entry is added to retrieved knowledge. This marginal reward favors entries that improve execution beyond what the bank already provides. We also reward task success with each entry alone to encourage individual utility. These signals jointly train the curator, whose generated knowledge is retrieved to guide a frozen executor.

We evaluate whether learned knowledge improves a frozen executor on ALFWorld, a household task environment (Shridhar et al., 2020), and WebShop, an online shopping environment (Yao et al., 2022a). Before each task, we retrieve $k = 10$ knowledge entries from the bank using the task query and provide them to the frozen executor as guidance. Knowledge Weaver achieves success rates of 54.0% and 42.0%, respectively. These results exceed task-success-only GRPO by 16.9 and 18.7 percentage points under the same executor. These gains demonstrate improved task performance and support a curation strategy aimed at reducing redundant overlap among knowledge entries.

Our contributions are summarized as follows.

- **Redundancy in knowledge curation.** We identify a limitation of GRPO using only task success as reward, which can produce general guidance with substantial informational redundancy across knowledge entries.
- **MI-guided curator training.** We propose Knowledge Weaver, which combines token-level information feedback with execution rewards to train useful and complementary knowledge generation.
- **Improved downstream performance.** On ALFWorld and WebShop, Knowledge Weaver improves SR over task-reward-only GRPO by 16.9 and 18.7 percentage points at $k = 10$, respectively. Its banks also exceed human-written banks on the reported aggregate metrics.

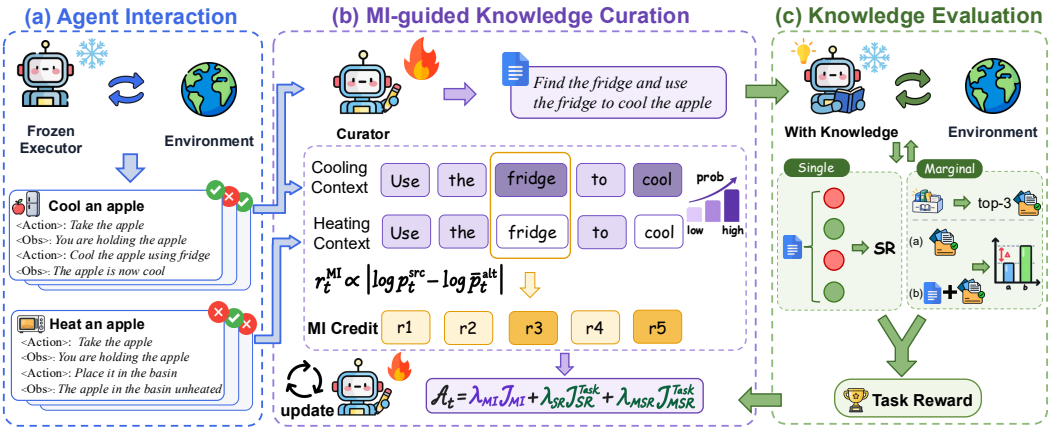


Figure 2: Overview of Knowledge Weaver. Token-level MI feedback promotes experience-dependent knowledge, while execution rewards assess individual utility and marginal gains over retrieved knowledge. Only the curator is trained and the executor remains frozen.

2 Related Work

Knowledge Construction from Experience. LLM agents construct reusable knowledge through memory organization, experience distillation, and skill synthesis. Memory systems extract and organize information for later retrieval (Chhikara et al., 2025; Xu et al., 2026), while experience distillation produces reflective lessons and generalizable strategies (Shinn et al., 2023; Zhao et al., 2024; Majumder et al., 2023; Ouyang et al., 2026). Structured representations include conditional guidelines (Fu et al., 2024; Chen et al., 2024), reusable workflows (Wang et al., 2024), and executable skills (Wang et al., 2023; Yuan et al., 2024; Zheng et al., 2025). AutoSkill (Yang et al., 2026) maintains skills derived from interaction traces, while Trace2Skill (Ni et al., 2026) consolidates trajectory-level lessons into transferable guidance. Skill-Pro (Mi et al., 2026) uses execution feedback to generate and verify procedural skills without parameter updates. RL-based approaches further learn to use and construct reusable knowledge. Skill0 (Lu et al., 2026) trains agents to internalize supplied skills, while Skill0.5 (Zhu et al., 2026) combines general skill internalization with task-specific skill utilization. SkillRL (Xia et al., 2026) couples skill distillation and library evolution with executor training. Skill1 (Shi et al., 2026) jointly learns skill selection, utilization, and distillation, while SkillRise (Yao et al., 2026) assigns curation credit through subsequent execution outcomes. Knowledge Weaver complements execution feedback with explicit information feedback to train knowledge generation while keeping the executor fixed.

Mutual Information. Mutual information (MI) provides a general criterion for learning informative representations and encouraging dependence between related variables. In representation learning, Contrastive Predictive Coding captures information predictive of future observations, while Deep InfoMax promotes dependence between inputs and learned representations (Oord et al., 2018; Hjelm et al., 2018). InfoGAN uses MI between latent codes and generated observations to encourage interpretable factors of variation (Chen et al., 2016). In reinforcement learning, DIAYN maximizes MI between skill identities and visited states to discover distinguishable behaviors (Eysenbach et al., 2018). In dialogue generation, MI-based objectives discourage generic responses and favor responses that convey information about the conversational context (Li et al., 2016; Zhang et al., 2018). AMI further models source-target dependence through adversarial optimization of forward and backward generation networks (Pan et al., 2020). For LLM agents, RAGEN-2 introduces MI-inspired proxies to assess the dependence of generated reasoning on input prompts (Wang et al., 2026). In knowledge curation, input dependence concerns whether generated entries retain the distinct information carried by their source trajectories. Knowledge Weaver applies token-level MI-inspired feedback during curator training to encourage experience-specific content.

3 METHOD

3.1 PRELIMINARIES

As shown in Figure 2, a trainable curator π_θ generates a knowledge entry $z \sim \pi_\theta(\cdot \mid x)$ from a context x of trajectories sampled from a pool $\mathcal{T}$, each recording a task's observations and actions. Training maintains an initially empty bank $\mathcal{B}$, expanded between updates. After training, the curator generates a separate evaluation bank $\mathcal{C} = \{z^{(i)}\}_{i=1}^{K}$ of valid entries from sampled contexts. A frozen executor uses a knowledge set S to solve task q. Writing its binary success as $s(q, S)$, we measure empirical utility on a finite reward-task set $\mathcal{D}$ by

$$U(S) = \frac{1}{|\mathcal{D}|} \sum_{q \in \mathcal{D}} s(q, S). \tag{1}$$

During evaluation, $S = \mathcal{R}_k(q, \mathcal{C})$ contains at most k entries retrieved once from the initial query and held fixed throughout execution.

3.2 EFFECTIVE KNOWLEDGE ACCUMULATION

Effective accumulation requires both individual utility and contributions beyond existing knowledge. With two equally weighted task types, a routine for an already-supported type and one for an uncovered type can be equally useful alone, yet only the latter extends the bank's capabilities.

We formalize this distinction with a nonnegative, integrable utility profile $f_i(u)$ for each entry $z^{(i)}$ over an idealized task coordinate $u \in \mathbb{R}$. The profile describes where and how strongly an entry helps; its area represents total individual utility. To avoid counting the same available utility repeatedly, bank coverage takes the best profile at each location,

$$\mathcal{E}(\mathcal{C}) = \int_{\mathbb{R}} \max_{1 \le i \le K} f_i(u)\, du. \tag{2}$$

Thus, repeated profiles add no coverage. This model assumes ideal selection of an entry without retrieval errors or interactions between entries; it illustrates utility overlap without predicting execution success. For a tractable comparison, use Gaussian profiles with fixed centers $c_1 < \cdots < c_K$, $K \ge 2$, common height $h > 0$, and width $\sigma > 0$,

$$f_i(u; h, \sigma) = h \exp\left(-\frac{(u - c_i)^2}{2\sigma^2}\right). \tag{3}$$

The center locates peak utility, while height and width describe its strength and extent. Each entry has total utility $A = \sqrt{2\pi}\sigma h$; write the resulting coverage as $\mathcal{E}(h, \sigma)$.

Proposition 1 (Utility scaling)**.** *For fixed centers and width, $\mathcal{E}(h, \sigma) = h\mathcal{E}(1, \sigma)$ is strictly increasing in $h > 0$.*

To compare overlap at equal individual utility, we instead fix each profile's area, so height must decrease as width grows.

Proposition 2 (Coverage under a fixed utility budget)**.** *For fixed centers and per-entry utility $A > 0$, let $h_A(\sigma) = A/(\sqrt{2\pi}\sigma)$. Then $\mathcal{E}_A(\sigma) = \mathcal{E}(h_A(\sigma), \sigma)$ strictly decreases with $\sigma > 0$.*

The first result isolates stronger individual utility. In the second, total individual utility remains KA, but broader profiles overlap more and yield less bank coverage. Narrowing profiles at fixed height need not improve coverage. A new entry adds coverage only where its utility exceeds the existing envelope (Figure 1). Appendix A gives both proofs and the marginal-coverage identity.

Why standalone-success GRPO Fails. Group Relative Policy Optimization (GRPO) (Shao et al., 2024) samples $G \ge 2$ candidates $z^{(g)}$ from a curator snapshot $\pi_{\text{old}}(\cdot \mid x)$ for the same source context x. Under outcome supervision, every token t in candidate g shares an advantage, the relative learning signal for the policy update,

$$A_{g,t}^{\text{GRPO}} = \frac{R_g - \bar{R}}{\sigma_R + \epsilon_{\text{num}}}, \tag{4}$$

where R_g is the candidate reward, $\bar{R}$ and σ_R are its group's reward mean and standard deviation, and $\epsilon_{\text{num}} > 0$ ensures numerical stability.

With standalone success $R_g = U(\{z^{(g)}\})$, this advantage contains no bank information. An already-covered routine receives positive advantage when $R_g > \bar{R}$, with no preference for an equally successful complementary entry. Shared outcome credit also treats source-specific and generic tokens alike. We therefore add bank-relative gains and local, source-dependent credit. We therefore seek to retain the distinct information carried by each input experience while avoiding additions that repeat what the bank already provides. These goals motivate two complementary signals. Marginal success feedback rewards contributions beyond existing knowledge (Section 3.3), while MI-inspired token credit encourages retention of source-dependent information (Section 3.4).

3.3 Marginal Success Feedback

Marginal success-rate (MSR) feedback encourages additions that improve execution beyond what the bank already supports. For each task $q \in \mathcal{D}$, retrieve up to m entries $B_q = \mathcal{R}_m(q, \mathcal{B})$ from a fixed training-bank snapshot and compute

$$\text{MSR}(z) = \frac{1}{|\mathcal{D}|} \sum_{q \in \mathcal{D}} \left[s(q, B_q \cup \{z\}) - s(q, B_q) \right]. \tag{5}$$

We also evaluate standalone success $U(\{z\})$, the success rate given only that entry, while MSR measures change relative to retrieved guidance. For source context x_i, candidates and all three conditions (standalone, bank-only, and bank-plus-candidate) share the reward-task set $\mathcal{D} = \mathcal{D}_i$, with B_q fixed per task. The candidate is appended without replacing retrieved entries. An empty bank gives a no-knowledge baseline. Unlike ideal coverage, execution gains can be negative under interference. They operationalize incremental utility without estimating the coverage integral.

Between updates, eligible valid entries absent from the bank are ranked by positive MSR and admitted under a quota. If none has positive gain, the bank is unchanged. The updated bank is used in the next training batch; Algorithm 1 gives the schedule.

3.4 MI-Inspired Information Feedback

Marginal success feedback evaluates what an entry contributes beyond the existing bank. We complement this signal with information feedback that encourages the curator to retain the distinct information contained in its input experience. Mutual information, $I(X; Z) = H(Z) - H(Z \mid X)$, with H denoting entropy, motivates measuring how source context X changes uncertainty about generated knowledge Z (Cover et al., 1991). For an entry $z = (z_1, \ldots, z_T)$ generated from x_i among $N \geq 2$ contexts, we contrast token probabilities under source and alternative contexts,

$$\delta_t = \log \frac{(N-1)\pi_{\text{old}}(z_t \mid x_i, z_{<t})}{\sum_{\ell \neq i} \pi_{\text{old}}(z_t \mid x_\ell, z_{<t})}, \quad r_t^{\text{MI}} = b_t \min\left\{c_{\text{MI}}, \lambda_{\text{MI}} |\delta_t|\right\}. \tag{6}$$

The token and its prefix are held fixed when comparing source likelihood with the mean likelihood under alternative contexts. We take the absolute value to reward the magnitude of input dependence, whether the source context increases or decreases a token's likelihood. Generic tokens with similar probabilities across contexts have small $|\delta_t|$ and receive little credit, whereas experience-specific tokens with larger contrasts receive more. The mask b_t limits this credit to knowledge-body tokens, with scale λ_{MI} and cap c_{MI}. Assigning credit locally favors input-specific content over generic wording, encouraging the curator to retain more information from each source experience.

Joint feedback and policy update. We construct the joint feedback by extending the SR-only task reward used in our GRPO baseline. SR rewards an entry's standalone utility without accounting for overlap with existing knowledge. We first add MI-inspired token credit to encourage preservation of information specific to each source experience. Because downstream evaluation concerns the utility of a knowledge bank, we further incorporate MSR to reward contributions beyond its existing entries. A formatting term additionally rewards valid structure and penalizes malformed outputs. Retaining GRPO's group-relative normalization (Equation 4), we combine these signals in a single

token-level advantage,

$$A_{g,t} = \frac{\lambda_{\mathrm{SR}} U(\{z^{(g)}\}) + \lambda_{\mathrm{MSR}} \mathrm{MSR}(z^{(g)}) + r_{\mathrm{fmt}}(z^{(g)}) - \bar{R}_{\mathrm{exec}}}{\sigma_{\mathrm{exec}} + \epsilon_{\mathrm{num}}} + r^{\mathrm{MI}}_{g,t}, \tag{7}$$

where $\bar{R}_{\mathrm{exec}}$ and σ_{exec} are the within-group mean and standard deviation of the combined SR, MSR, and formatting reward. The weights λ_{SR} and λ_{MSR} control the two utility tenot MI or inter-entry redundancy directlyrms. MSR augments GRPO's reward comparison with each entry's contribution to the bank, while MI supplies local credit for source-dependent content. Invalid candidates receive only the formatting penalty and no MI credit. We retain GRPO for policy optimization, using the combined feedback to update only the curator through its clipped objective (Shao et al., 2024). Optimization details are provided in Appendix F.1.

4 Experiments

4.1 Experimental Setup

Environments. We evaluate our method on two benchmarks: ALFWorld (Shridhar et al., 2020) and WebShop (Yao et al., 2022a). ALFWorld is a text-based environment in which an agent completes household tasks through sequential natural-language interactions. We evaluate knowledge banks on the `valid_unseen` split across six canonical task types: Pick, Clean, Heat, Cool, Look, and Pick Two. We report the success rate (SR) for each task type and across all evaluation tasks. WebShop is a simulated web shopping environment in which an agent searches for products, navigates product pages, and purchases an item that satisfies a user's requirements. We sample 100 tasks from the test set to evaluate the knowledge banks and report both the environment score and SR.

Training and Evaluation Protocol. Throughout training, we freeze the executor and optimize only the curator. We assign equal weights of 0.5 to standalone SR and marginal SR. Token-level MI credit is scaled by 0.1 and capped at 0.15 after scaling. We use Qwen3.5-4b as the training model. After training, we use the trained curator to construct knowledge banks in a separate generation stage. We then evaluate the resulting banks with the frozen executor. Each knowledge bank contains 50 entries(except for SkillRL), with each entry limited to 100 words. To generate each entry, we randomly sample four trajectories from a trajectory pool. For each evaluation task, we retrieve the top-k entries using the `all-mpnet-base-v2` encoder (Reimers & Gurevych, 2019), ranking entries by their cosine similarity to the initial task query. Retrieval occurs once, before the first action, and the selected entries remain fixed throughout the episode. We use $k = 10$ for all banks in the main comparison and evaluate every $k \in \{0, \ldots, 15\}$ on in the scaling analysis.

Baselines. We compare four categories of knowledge banks, following Table 1: (1) *LLM Curators*: banks generated directly by Qwen3.5-4B, GPT-5 (OpenAI, 2025), Claude Opus 4.8 (Anthropic, 2026), and Gemini 3.7 Flash (Google DeepMind, 2026), (2) *Established Knowledge Banks*: publicly released banks from SkillRL (Xia et al., 2026) and human-written banks refined by experts, (3) *RL-Trained*: GRPO (Shao et al., 2024) baseline, and SkillRise (Yao et al., 2026) and Skill1 (Shi et al., 2026), which use execution feedback to improve knowledge generation and refinement, (4) *Prompt-Based*: ExpeL (Zhao et al., 2024), which iteratively extracts and refines insights from successful and failed trajectories, and Mem0 (Chhikara et al., 2025), which extracts and organizes memories for later retrieval. We also include zero-shot execution (k=0) as a reference without external knowledge.

4.2 Main Results

Improved Performance across Benchmarks. Table 1 shows that our method achieves the highest values on all three aggregate metrics with the same frozen executor at $k = 10$. Overall SR reaches 54.0% on ALFWorld and 42.0% on WebShop, exceeding GRPO by 16.9 and 18.7 percentage points. On WebShop, its mean environment score reaches 66.4, compared with 49.0 for GRPO, indicating higher average reward alongside improved task success. Our method also outperforms SkillRise and Skill1 on both benchmarks. It exceeds SkillRL, the strongest ALFWorld baseline, by 2.8 percentage points in overall SR. Human-written banks achieve the highest baseline mean environment score on WebShop, but our method surpasses them on all three aggregate metrics. Gains over GRPO span

Table 1: Performance of knowledge banks with a frozen Qwen3.5-4B executor on ALFWorld and WebShop. Each bank supplies $k = 10$ retrieved entries per task and zero-shot execution uses no external knowledge. We report success rate (SR, %) and mean environment score on WebShop. All results are averaged over 3 seeds, with standard deviations shown as subscripts for the Overall and WebShop columns. **Bold** and underlining mark the best and second-best in each column.

	ALFWorld (SR%)							**WebShop**	
Method	Pick	Clean	Heat	Cool	Look	Pick Two	Overall	Score	SR%
Zero-shot	30.6	10.8	31.9	30.2	24.1	25.5	$24.6_{\pm 1.5}$	$53.5_{\pm 1.9}$	$27.3_{\pm 1.5}$
LLM CURATORS									
Qwen3.5-4B	40.3	43.0	24.6	41.3	48.1	15.7	$36.3_{\pm 3.0}$	$40.6_{\pm 2.2}$	$14.7_{\pm 2.1}$
GPT-5	40.3	39.8	27.5	61.9	50.0	21.6	$40.3_{\pm 7.9}$	$45.2_{\pm 6.8}$	$18.7_{\pm 7.4}$
Claude Opus 4.8	40.3	30.1	40.6	36.5	72.2	3.9	$37.1_{\pm 6.2}$	$55.2_{\pm 2.5}$	$24.7_{\pm 0.6}$
Gemini 3.7 Flash	25.0	11.8	18.8	25.4	64.8	11.8	$24.6_{\pm 6.4}$	$61.3_{\pm 5.4}$	$35.7_{\pm 5.9}$
ESTABLISHED KNOWLEDGE BANKS									
SkillRL	34.7	**50.5**	50.7	**68.3**	66.7	**39.2**	$51.2_{\pm 2.4}$	$36.6_{\pm 3.0}$	$15.7_{\pm 0.6}$
Human-written	**47.2**	48.4	**63.8**	52.4	55.6	25.5	$49.5_{\pm 1.1}$	$63.1_{\pm 1.5}$	$32.0_{\pm 1.0}$
RL-TRAINED									
GRPO	30.6	26.9	34.8	44.4	55.6	**39.2**	$37.1_{\pm 3.7}$	$49.0_{\pm 4.8}$	$23.3_{\pm 2.1}$
SkillRise	43.8	22.6	32.6	28.6	47.2	29.4	$33.2_{\pm 1.6}$	$40.0_{\pm 5.6}$	$18.7_{\pm 3.5}$
Skill1	37.5	25.8	14.5	34.9	44.4	25.5	$29.9_{\pm 8.2}$	$38.5_{\pm 0.9}$	$17.0_{\pm 1.0}$
PROMPT-BASED									
ExpeL	38.9	18.3	23.2	27.0	37.0	31.4	$28.4_{\pm 2.7}$	$58.4_{\pm 1.2}$	$24.7_{\pm 2.1}$
Mem0	36.8	37.1	52.2	47.6	47.2	25.5	$41.2_{\pm 0.9}$	$50.1_{\pm 1.3}$	$19.0_{\pm 2.0}$
Our	45.8	48.4	**63.8**	54.0	**87.0**	27.5	$\mathbf{54.0}_{\pm 0.4}$	$\mathbf{66.4}_{\pm 2.6}$	$\mathbf{42.0}_{\pm 2.0}$

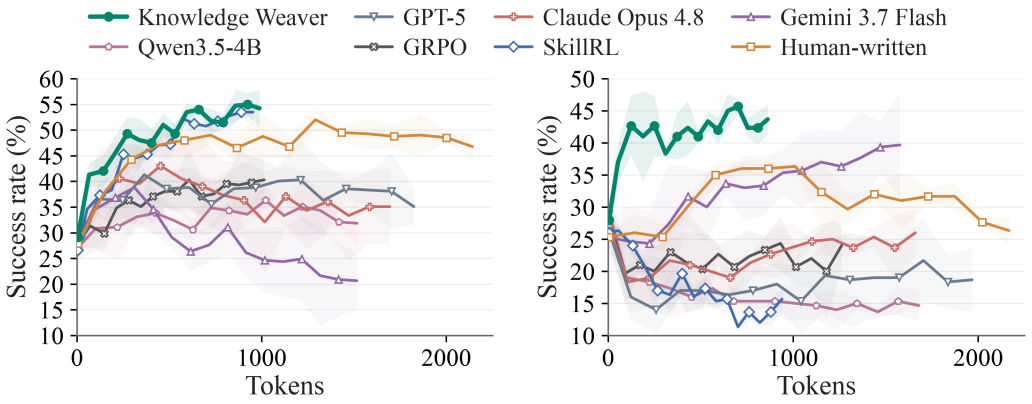
five of six ALFWorld task types, although performance on Pick Two remains lower. These results support training knowledge curators with feedback beyond standalone task success.

Strong Base Models Do Not Guarantee Useful Knowledge. In Table 1, we evaluate strong general-purpose models as knowledge curators with the same frozen executor. Our trained Qwen3.5-4B curator outperforms GPT-5, Claude Opus 4.8, and Gemini 3.7 Flash on all three aggregate metrics. On ALFWorld, GPT-5 and Claude banks achieve overall SRs of 40.3% and 37.1%, respectively, above the zero-shot SR of 24.6%. On WebShop, however, their banks yield SRs of 18.7% and 24.7%, below the zero-shot SR of 27.3%. Gemini achieves the highest WebShop SR among these general-purpose curators (35.7%), whereas its overall ALFWorld SR matches the zero-shot mean (24.6%). These results show that strong general capabilities do not necessarily translate into knowledge that benefits downstream execution.

4.3 KNOWLEDGE BANK QUALITY ANALYSIS

Useful Knowledge Requires Relatively Little Context. Figure 3 examines whether the maintable gains persist when the number of retrieved entries changes. Knowledge Weaver achieves the highest mean WebShop SR among the eight plotted methods at every positive retrieval count. At $k = 10$, its knowledge context averages approximately 660 tokens on ALFWorld and 594 on WebShop, less than half the corresponding human-written context lengths. It nevertheless achieves higher SR on both benchmarks. The advantage therefore accompanies compact guidance, rather than a larger volume of injected text. As retrieval increases from $k = 9$ to 15, SR remains within 51.5-55.0% on ALFWorld and 42.0-45.7% on WebShop. The gains persist across this range despite fluctuations, reducing dependence on a single retrieval setting. ALFWorld performance is less uniformly dominant: human-written banks perform better at several smaller retrieval counts, and SkillRL slightly leads at $k = 12$.

Additional Knowledge Can Help or Hinder Execution. Increasing retrieval exposes clear differences in how banks support the executor. On ALFWorld, Gemini's SR declines from 38.8% at $k = 3$ to 20.6% at $k = 15$, whereas Knowledge Weaver largely retains its gains. On WebShop, Qwen3.5-4B, GPT-5, Claude, and GRPO remain below their own no-knowledge controls at every



(a) ALFWorld (b) WebShop

Figure 3: Success rate as the number of retrieved knowledge entries k varies from 0 to 15. The horizontal axis reports the number of tokens in the formatted knowledge block injected into the frozen executor, averaged over tasks and runs. Lines and shading show the mean and one standard deviation across three runs. Each curve uses its own no-knowledge control at $k = 0$.

positive retrieval count. Their additional context therefore fails to recover the performance lost when guidance is introduced. This pattern is consistent with unhelpful or conflicting guidance, although the curves alone cannot distinguish interference from retrieval mismatch or other content effects. Effective accumulation consequently requires assessing how entries work together, as well as whether individual entries appear useful. This observation motivates the marginal-utility objective and its contribution is examined alongside MI in the ablation. These sweeps retrieve from fixed banks, with observed token lengths rather than matched token budgets. To complement these execution trends, we quantify whether different knowledge entries tend to be retrieved for the same tasks. At $k = 10$, we compute mean pairwise Jaccard overlap between these task sets for entries retrieved at least once, then average over three banks per method. Overlap is lower for Knowledge Weaver than for GRPO, with values of 12.92% versus 13.58% on ALFWorld and 15.35% versus 24.48% on WebShop. Thus, sustained execution gains with longer context coexist with lower retrieval overlap at $k = 10$. These observations motivate our joint use of MI feedback and marginal success rewards to preserve useful information, with execution benefits examined in Section 4.4.

4.4 Ablation Studies

MI and MSR Improve Knowledge Utility. Table 2 compares SR-only training with two progressively augmented objectives, SR + MI and SR + MI + MSR. The first extension adds token-level MI, increasing ALFWorld SR from 37.1% to 46.0% and WebShop SR from 23.3% to 40.7%. WebShop score also improves from 49.0 to 65.8, supporting information feedback as a useful addition to standalone-success training.

Building on SR + MI, the full objective adds MSR to reward contributions beyond existing knowledge. ALFWorld SR increases further to 54.0%, an additional 8.0 percentage points, comparable to the 8.9-point improvement from adding MI. On WebShop, SR reaches 42.0% and score reaches 66.4, with a smaller additional SR gain of 1.3 points. Both extensions therefore improve performance, with similarly sized SR gains on ALFWorld and a larger gain from MI on WebShop.

Table 2: Reward ablation at $k = 10$ with a frozen Qwen3.5-4B executor. SR rewards individual task success, MSR rewards success-rate gains over retrieved knowledge, and MI provides token-level information feedback. Formatting rewards are shared across configurations. Values are three-run means under different seeds. **bold** marks column best.

Rewards	ALFWorld	WebShop	
	SR	Score	SR
SR	37.1	49.0	23.3
SR + MI	46.0	65.8	40.7
SR + MI + MSR	**54.0**	**66.4**	**42.0**

Knowledge Banks Transfer across Frozen Executors. We reuse banks generated by a curator trained with feedback from a frozen Qwen3.5-4B executor. Table 3 reports their performance with eight additional frozen executors at $k = 8$. Knowledge Weaver improves ALFWorld SR and both WebShop metrics over no-knowledge execution for all eight executors. Relative to GRPO, it achieves higher ALFWorld SR and WebShop scores for seven of eight executors on each metric, and higher WebShop SR for six, with one tie. GRPO banks improve ALFWorld SR for all eight executors but reduce WebShop SR for four. These results support reusing Knowledge Weaver banks across the evaluated executors without retraining the curator for each target model.

Table 3: Transfer of knowledge banks across frozen executors. None denotes execution without knowledge ($k = 0$), while GRPO and Knowledge Weaver (Ours) retrieve $k = 8$ entries per task. Knowledge-augmented results are averaged over three banks per method. **Bold** marks the best result for each executor and metric.

Executor	**ALFWorld SR (%)**			**WebShop Score**			**WebShop SR (%)**		
	None	GRPO	Ours	None	GRPO	Ours	None	GRPO	Ours
Qwen3.5-4B	24.6	38.1	**49.3**	53.5	49.0	**67.5**	27.3	20.7	**41.0**
Qwen3.5-27B	48.5	**61.9**	61.2	54.9	55.6	**66.9**	25.0	23.0	**41.7**
Llama-3-8B-Instruct	6.7	7.2	**12.7**	2.7	10.3	**10.5**	0.0	**4.7**	4.0
Llama-3.1-8B-Instruct	9.7	14.9	**21.1**	16.0	6.7	**19.7**	6.0	2.3	**11.7**
Phi-4-mini-instruct	7.5	9.5	**13.9**	2.27	**8.9**	5.4	1.0	**3.3**	**3.3**
Phi-4	19.4	32.6	**38.1**	57.0	51.3	**65.2**	13.0	16.7	**33.3**
DeepSeek-V4-Flash	56.0	64.2	**64.9**	57.0	56.2	**69.2**	30.0	26.0	**46.3**
MiniMax-M2.5	35.1	53.7	**56.5**	62.8	64.8	**73.6**	30.0	33.0	**48.0**
MiniMax-M2.1	70.9	73.9	**74.4**	73.2	68.7	**75.8**	43.0	37.0	**49.7**

5 CONCLUSION

We introduced Knowledge Weaver, an RL framework that trains knowledge curators to preserve information through token-level MI feedback and rewards for marginal gains over retrieved knowledge, while rewarding standalone success. On ALFWorld and WebShop, the resulting banks outperform the evaluated baselines with a frozen executor, and the gains hold across retrieval sizes. Ablation indicates that MI adds value beyond task success alone. The banks also transfer to other executors, though the gains vary by model. These results support information preservation as a useful principle for building reusable knowledge from agent experience.

We plan several directions that follow from this work in the future. First, because our experiments keep the executor frozen, a natural extension is to jointly train the curator and executor, allowing knowledge construction and utilization to improve in tandem. Second, the uneven transfer gains suggest that knowledge effective for one executor may not transfer directly to another, motivating the development of knowledge banks that generalize across models. Finally, extending Knowledge Weaver to longer-horizon tasks, where the bank is continually updated as experience accumulates, would provide a stronger test of how well knowledge preservation scales over time. More broadly, we view learning to curate reusable knowledge as a step toward agents that self-improve continually through their own experience.

AI USE STATEMENT

In this work, we used generative AI tools to assist with the implementation of software code and the analysis and interpretation of experimental results. We did not use generative AI tools to formulate the core research ideas, hypotheses, or methodological design of this work.

Additionally, we used generative AI tools to improve the readability and language of the manuscript and to create or refine minor visual elements in some figures.

All AI-assisted code was reviewed and tested by the authors, and all AI-assisted analyses of experimental results were independently verified against the original experimental outputs. All AI-assisted text and visual elements were reviewed and revised by the authors. We take responsibility for the

final content of this work, including text, claims, code, analyses, and artifacts produced with the aid of generative AI.

Reproducibility statement

Section 3 defines the feedback signals and training objective. Section 4.1 describes the ALFWorld and WebShop evaluation protocol. Appendix C provides the prompts used for knowledge generation and execution. Appendix E reports the hyperparameter settings in Table 4. Appendix F presents the training pseudocode and GRPO update. Appendix D explains how the baselines were implemented and adapted to the shared evaluation protocol.

Ethics statement

Human volunteers wrote knowledge entries from sampled agent trajectories to construct the Human-written baseline. Participation was voluntary, and informed consent was obtained from all volunteers.

References

Anthropic. Claude Opus 4.8 System Card. System card, 2026. URL https://www.anthropic.com/claude-opus-4-8-system-card.

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

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29, 2016.

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.

Thomas M Cover, Joy A Thomas, and John Kieffer. *Elements of information theory*, volume 2. wiley New York, 1991.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

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

Google DeepMind. Gemini 3.7 Flash Model Card. Model card, August 2026. URL https://deepmind.google/models/model-cards/gemini-3-7-flash/.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. In *International Conference on Learning Representations*, volume 2024, pp. 52690–52717, 2024.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and William B Dolan. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 110–119, 2016.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, et al. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 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.

Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Peter Jansen, Oyvind Tafjord, Niket Tandon, Li Zhang, Chris Callison-Burch, and Peter Clark. Clin: A continually learning language agent for rapid task adaptation and generalization. *arXiv preprint arXiv:2310.10134*, 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.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

OpenAI. GPT-5 System Card. System card, 2025. URL `https://openai.com/index/gpt-5-system-card/`.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, volume 2026, pp. 94327–94354, 2026.

Boyuan Pan, Yazheng Yang, Kaizhao Liang, Bhavya Kailkhura, Zhongming Jin, Xian-Sheng Hua, Deng Cai, and Bo Li. Adversarial mutual information for text generation. In *International Conference on Machine Learning*, pp. 7476–7486. PMLR, 2020.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

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

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

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

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7080–7106, 2026.

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.

Zihan Wang, Chi Gui, Xing Jin, Qineng Wang, Licheng Liu, Kangrui Wang, Shiqi Chen, Linjie Li, Zhengyuan Yang, Pingyue Zhang, et al. Ragen-2: Reasoning collapse in agentic rl. *arXiv preprint arXiv:2604.06268*, 2026.

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

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*, 2026.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604, 2026.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

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

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022b.

Zhiyuan Yao, Yuxin Chen, Zhengxi Lu, Zishan Xu, Yueqing Sun, Yifu Guo, Yuquan Lu, Zhengzhou Cai, Kangning Zhang, Zhuowen Han, et al. Skillrise: Agentic reinforcement learning for cross-task skill evolution. *arXiv preprint arXiv:2607.26784*, 2026.

Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi Fung, Hao Peng, and Heng Ji. Craft: Customizing llms by creating and retrieving from specialized toolsets. In *International Conference on Learning Representations*, volume 2024, pp. 40097–40125, 2024.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, volume 2026, pp. 86069–86100, 2026.

Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and Bill Dolan. Generating informative and diverse conversational responses via adversarial information maximization. *Advances in Neural Information Processing Systems*, 31, 2018.

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.

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

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *International Conference on Learning Representations*, volume 2024, pp. 19036–19066, 2024.

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

A PROOFS

We use the Gaussian utility profiles from Equation 3, with $K \geq 2$ fixed centers $c_1 < \cdots < c_K$ and common height $h > 0$ and scale $\sigma > 0$. Specializing Equation 2 gives the envelope coverage $\mathcal{E}(h, \sigma) = \int_{\mathbb{R}} \max_i f_i(u; h, \sigma)\, du$. Write $\Delta_i = c_{i+1} - c_i > 0$ for the adjacent gaps, and let ϕ and Φ denote the standard normal density and cumulative distribution function.

A.1 PROOF OF PROPOSITION 1

Proof. For fixed σ, changing the common height rescales every profile by the same positive factor. This factor can be taken outside the pointwise maximum and the integral, giving

$$\mathcal{E}(h, \sigma) = h \int_{\mathbb{R}} \max_{1 \leq i \leq K} \exp\left(-\frac{(u - c_i)^2}{2\sigma^2}\right) du = h\, \mathcal{E}(1, \sigma). \tag{8}$$

To establish strict monotonicity, it remains to check that the coefficient $\mathcal{E}(1, \sigma)$ is positive and finite. The envelope is bounded below by any individual profile and above by their sum. Since each unit-height profile integrates to $\sqrt{2\pi}\,\sigma$, these bounds imply

$$0 < \sqrt{2\pi}\,\sigma \leq \mathcal{E}(1, \sigma) \leq K\sqrt{2\pi}\,\sigma < \infty.$$

Thus coverage is linear and strictly increasing in the common height h. □

A.2 PROOF OF PROPOSITION 2

Proof. Fix the per-entry mass $A > 0$. The Gaussian integral gives $A = \sqrt{2\pi}\,\sigma h$, so the height must vary with σ according to

$$h_A(\sigma) = \frac{A}{\sqrt{2\pi}\,\sigma}, \qquad f_i(u; h_A(\sigma), \sigma) = \frac{A}{\sigma}\phi\left(\frac{u - c_i}{\sigma}\right). \tag{9}$$

Each profile therefore has mass A for every $\sigma > 0$. We write $\mathcal{E}_A(\sigma) = \mathcal{E}(h_A(\sigma), \sigma)$ for the coverage under this constraint.

Because all profiles share the same height and scale, the nearest center provides the largest utility at each location. Define the adjacent midpoints $m_i = (c_i + c_{i+1})/2$ for $i = 1, \ldots, K-1$, with $m_0 = -\infty$ and $m_K = +\infty$. On $[m_{i-1}, m_i]$, the center c_i is nearest to u, so f_i attains the upper envelope. We can therefore integrate the envelope one interval at a time:

$$\begin{aligned}\mathcal{E}_A(\sigma) &= \sum_{i=1}^{K} \int_{m_{i-1}}^{m_i} f_i(u; h_A(\sigma), \sigma)\, du \\ &= A \sum_{i=1}^{K} \left[\Phi\left(\frac{m_i - c_i}{\sigma}\right) - \Phi\left(\frac{m_{i-1} - c_i}{\sigma}\right)\right].\end{aligned} \tag{10}$$

For each interior interval, the upper and lower bounds relative to its center are $\Delta_i/2$ and $-\Delta_{i-1}/2$. Using $\Phi(-z) = 1 - \Phi(z)$, its contribution is $A[\Phi(\Delta_i/(2\sigma)) + \Phi(\Delta_{i-1}/(2\sigma)) - 1]$. The two end intervals contribute $A\Phi(\Delta_1/(2\sigma))$ and $A\Phi(\Delta_{K-1}/(2\sigma))$, respectively. Thus each adjacent gap contributes twice, while each of the $K - 2$ interior intervals contributes one subtraction of A. Summing gives

$$\mathcal{E}_A(\sigma) = A\left[2\sum_{i=1}^{K-1} \Phi\left(\frac{\Delta_i}{2\sigma}\right) - (K - 2)\right]. \tag{11}$$

This expression also holds for $K = 2$, when there are no interior intervals. Differentiating with A and the centers fixed, and using $\Phi' = \phi$, yields

$$\frac{d\mathcal{E}_A(\sigma)}{d\sigma} = -\frac{A}{\sigma^2} \sum_{i=1}^{K-1} \Delta_i\, \phi\left(\frac{\Delta_i}{2\sigma}\right) < 0. \tag{12}$$

The sum contains at least one term because $K \geq 2$, and every term is positive because $\Delta_i > 0$ and ϕ is strictly positive. Hence coverage strictly decreases with σ under the fixed per-entry mass constraint, proving the proposition. □

Redundancy and marginal coverage. At each location, utility counted by the sum of profiles but omitted by their maximum represents redundant overlap. Its integral is $\mathcal{R}_A(\sigma) = \int_{\mathbb{R}} [\sum_i f_i(u; h_A(\sigma), \sigma) - \max_i f_i(u; h_A(\sigma), \sigma)]\, du$. Subtracting Equation 11 from the total mass KA gives

$$\mathcal{R}_A(\sigma) = KA - \mathcal{E}_A(\sigma) = 2A \sum_{i=1}^{K-1} \Phi\left(-\frac{\Delta_i}{2\sigma}\right). \tag{13}$$

Since $\mathcal{R}'_A = -\mathcal{E}'_A > 0$, increasing the common scale increases redundant overlap. As $\sigma \to 0^+$, each Φ term in Equation 11 tends to 1, giving coverage KA in the limit. As $\sigma \to \infty$, each term tends to $1/2$, giving limiting coverage A. Consequently, the coverage efficiency $\mathcal{E}_A/(KA)$ decreases from a limit of 1 to a limit of $1/K$.

More generally, for any nonnegative, integrable candidate profile f_z, its marginal coverage is

$$\begin{aligned} \Delta_{\mathrm{cov}}(z \mid \mathcal{C}) &= \mathcal{E}(\mathcal{C} \cup \{z\}) - \mathcal{E}(\mathcal{C}) \\ &= \int_{\mathbb{R}} \left[f_z(u) - \max_{1 \le i \le K} f_i(u) \right]_+ du, \end{aligned} \tag{14}$$

where $[a]_+ = \max(a, 0)$. This follows pointwise from $\max\{a, b\} - b = [a - b]_+$. Apply the identity with $a = f_z(u)$ and $b = \max_i f_i(u)$, then integrate. It holds for any nonnegative, integrable profiles and requires no Gaussian assumption.

For the Gaussian specialization, consider adding an entry with the same A and σ at $c_K + \Delta$, where $\Delta > 0$. The coverage formula gains one gap term and one additional subtraction of A, so the marginal coverage is $A[2\Phi(\Delta/(2\sigma)) - 1]$. It is strictly increasing in Δ and lies in $(0, A)$, approaching 0 as $\Delta \to 0^+$ and A as $\Delta \to \infty$. The same argument applies to an entry added beyond the leftmost center.

B Case Studies of Knowledge Bank Content

We inspect knowledge banks generated from step-40 curator checkpoints to examine what repeated and distinct entries contain. All examples below are complete, unedited entries from the GRPO and Knowledge Weave used in Table 1. Each source bank contains 50 entries. We select cases from the knowledge banks of GRPO and Knowledge Weaver.

B.1 ALFWorld

GRPO | Step 40

Possession before placement
locate your target object, verify it is on the surface with take, then immediately navigate to the destination and execute move to place it. Avoid navigating without picking up the object or performing unnecessary inventory checks that delay the action.

Possession before placement
always take the object from the source before moving or attempting to place it there, even if the action seems redundant, to ensure possession is maintained and progress is made.

Possession before placement
Check your inventory before moving to the destination to confirm you are carrying the correct object, then proceed directly to the destination and place the object there immediately without returning to the source.

Knowledge Weaver | Step 40

Cooling with the fridge
When cooling an object for placement, immediately pick it up from the countertop after navigating there, then head to the fridge and cool the object there, rather than returning to the countertop or searching for the object multiple times.

Heating with the microwave
If you need to heat an item before placing it: verify the item is accessible by picking it up first (e.g., from the countertop), then heat it in the microwave using valid actions like move and heat; this ensures the object is in your possession and eliminates the error of heating an object that is still sitting in one location while you navigate to gain access to another, which consumes steps without progressing toward success.

Inspection under a lamp
Before looking at an object in light, verify you are holding the object; if not, locate it by navigating to its current position and pick it up, then turn on the nearby lamp by using it only after the object is in your inventory.

The three GRPO entries reduce to the same possession-before-placement routine despite their different wording. The selected Knowledge Weaver entries retain distinct operations: cooling with a fridge, heating with a microwave, and activating a lamp while holding the object. These differences concern the prescribed action and appliance, rather than only an object-name substitution. They demonstrate task-specific content in individual entries, but do not establish low redundancy throughout the bank. Inventory checking remains a recurring motif in both source banks.

B.2 WEBSHOP

GRPO | Step 40

Verify color before purchase
Before clicking Buy Now, explicitly verify the visible "color" filter has been applied by selecting a specific color option, rather than relying on assumptions that a search query string initializes this setting.

Verify color before purchase
Before clicking the Buy Now button, explicitly verify that the selected color option has actually been applied by checking the instruction text for a confirmation message (e.g., "You have clicked dark gray") rather than relying on the absence of an error message. This prevents the "invisible update" failure where missing a confirmation causes the price filter to reset, allowing unselected colors to bypass the intended price constraint and wasting search steps.

Verify color before purchase
Before clicking Buy Now, verify the selected color filter option is actually selected on the product interface-based on the URL or page state, not solely on the search query or user intent.

Knowledge Weaver | Step 40

Select a product attribute
When a product page lists multiple color or size options alongside the current selection, always click the specific correct option to apply the attribute before clicking Buy Now.

Constrain the search query
When combined product attributes like color or material type are critical to the search intent, explicitly specify these color and material constraints in the search query rather than relying solely on the displayed product page for confirmation, as clicking a product page to select a specific color attribute often triggers an error stating "INVALID(click target is not currently visible)."

Continue to the next page
When a product page displays no purchasable option of the specified color or size, explicitly advance to the next page to find the correct attribute before clicking Buy Now, rather than leaving the system with a non-compliant selection field.

The GRPO examples provide general guidance on checking color before purchase, varying the suggested confirmation signal while repeating the same decision rule. Across this bank, 48 of 50 entries begin with “Before”. The selected Knowledge Weaver entries cover three different decisions: applying an available attribute, expressing constraints in the search query, and continuing to another page when a suitable option is absent.

C CURATOR AND EXECUTOR PROMPTS

We report the system and user message templates used for knowledge curation and execution on ALFWorld and WebShop. The curator receives four trajectories from the same task type and returns a single knowledge entry of at most 100 words, enclosed in `<skill>` and `</skill>` tags. We reproduce the evidence-conditioned prompts, including their output requirements, and retain the implementation’s term “skill.” The prompt wording is preserved; whitespace is adjusted for typesetting.

Fields such as `{trajectories}`, `{observation}`, and `{guidance}` represent content inserted at runtime. The trajectory field contains the rendered source histories, and the guidance field contains the knowledge supplied to the executor. For fixed-bank evaluation, multiple retrieved entries are numbered under “Retrieved guidance skills:” and remain fixed throughout the episode. The memory block is omitted when no prior interactions are available, and the WebShop invalid-response

line is included only when retrying an invalid response. In the ALFWorld action vocabulary, `{obj}`, `{recep}`, and `{appliance}` are literal action-template slots.

The executor fields are shown after preprocessing and before model-specific chat formatting. The recorded evaluations retain up to five recent interactions. ALFWorld supplies the final 1,800 observation characters and at most 140 admissible commands; WebShop limits the current observation to 5,000 characters and lists at most 60 visible click targets. Command and click-target lists are sorted before truncation.

C.1 ALFWORLD CURATOR

ALFWorld curator | System

You extract one useful guidance skill from agent trajectories. The acting agent already receives general operating guidance to consult its legal actions, navigate where needed, reveal accessible contents, and carry out the requested interaction. Treat that guidance as given rather than rewriting it as the learned skill.

Study the complete trajectories as behavioral evidence. Keep investigating beyond convenient surface explanations until the histories support an additional consequential insight that can improve future decisions. The conclusion must depend on what actually happened in the trajectories. Do not merely summarize the task or action-space description, imitate a fixed writing pattern, or invent unsupported details. Reason privately and output only the final natural-language skill required by the output contract. Return exactly one skill wrapped by <skill> and </skill>, with no text outside the tags.

ALFWorld curator | User

Task category
{category} ({category_note})

Legal action vocabulary
The agent may use these action templates: go to {recep}, open/close {recep}, take {obj} from {recep}, put {obj} in/on {recep}, use {obj}, heat/cool/clean {obj} with {appliance}, examine {obj}, look. This list only constrains which actions are legal. It is not trajectory evidence and must not be restated as the learned skill. The agent only sees the current room view. Receptacles must be navigated to, and some must be opened before their contents become visible.

Complete trajectories
{trajectories}

Skill content requirements
- Read across the complete action histories and discover a consequential issue or insight that is easy to miss from the task description alone, then give useful guidance for addressing it.
- Ground the guidance in differences, consequences, or lack of progress visible in the supplied trajectories. If the same advice could be written without reading them, investigate further.
- Do not mechanically recap the action sequence.
- Use only legal actions from the action vocabulary. Do not invent actions or unsupported facts.
- Write one compact natural-language paragraph of at most 100 words, without following a fixed rhetorical template.

Strict output requirements
- Return exactly: <skill><one natural-language guidance paragraph></skill>
- The opening tag must be the first non-whitespace text and the closing tag the last.
- Use exactly one opening tag and one closing tag; do not add analysis, markdown, labels, nested tags, or text outside the wrapper.

C.2 ALFWORLD EXECUTOR

ALFWorld executor | System

You are an ALFWorld TextWorld agent. Choose the next action using this exact format:
Thought: <concise reasoning, max 30 words>
Action: <copy exactly one command from the admissible command list>
Write the Action line immediately after the Thought line. Do not add any other text.
Guidance skill:
{guidance}

ALFWorld executor | User

Task: {task_description}
Observation: {observation}
Recent online episodic memory:
{online_memory}
Admissible commands:
{admissible_commands}

Respond with Thought and Action:

C.3 WEBSHOP CURATOR

WebShop curator | System

You extract one useful guidance skill from shopping-agent trajectories. Read the complete histories as behavioral evidence and begin with the concrete details that explain why an action helped, failed, or changed the outcome. Infer one non-obvious, consequential insight that would materially improve a future agent's decision in a similar situation. Let the evidence determine what the skill is about instead of starting from a predetermined workflow or restating advice that could be written without these histories.

Make the insight specific enough to guide a decision. Ground every claim in an observed state, action consequence, or outcome difference. Do not produce generic boilerplate, a reusable checklist, a trajectory recap, or unsupported product facts. Reason privately and output only the final skill required by the output contract.

Return exactly one <skill>...</skill> wrapper and no text outside it.

WebShop curator | User

Task category
WebShop product search, option selection, and purchase

Legal action vocabulary
The agent can call search[concise query] when a search box is present and click[exact visible target] for products, navigation controls, product options, and Buy Now. This vocabulary only constrains legal behavior; it is not trajectory evidence and must not be restated as the skill.

Complete trajectory evidence
{trajectories}

Skill content requirements

WebShop curator | User (continued)

- Extract a consequential insight whose usefulness depends on reading these histories closely, including evidence from failed progress, misleading page state, or a successful change in behavior.
- Prefer the smallest behaviorally meaningful correction or decision principle supported by the evidence. If the same advice could be written without the histories, investigate further.
- Let concrete trajectory evidence determine both the content and wording. Do not fall back on generic boilerplate, a reusable checklist, or a trajectory recap.
- Use only legal WebShop actions and facts supported by the supplied histories.
- Write one compact natural-language paragraph of at most 100 words.

Strict output requirements
- Return exactly: <skill><one natural-language guidance paragraph></skill>
- The opening tag must be the first non-whitespace text and the closing tag the last.
- Use exactly one opening tag and one closing tag. Do not add analysis, markdown, labels, nested tags, or text outside the wrapper.

C.4 WebShop executor

WebShop executor | System

You are a WebShop text agent. Select the next action to satisfy the shopping instruction and maximize the final WebShop reward.
Respond using exactly two lines:
Thought: <concise reasoning, at most 35 words>
Action: <one valid action>
Use search[concise keywords] only when search is available. Use click[exact target] by copying one currently visible clickable target. Do not add any other text.
Guidance skill:
{guidance}

WebShop executor | User

Shopping instruction: {shopping_instruction}
Current page observation:
{observation}
Recent online memory:
{online_memory}
Search action: {available_or_unavailable}
Visible click targets:
{visible_click_targets}
Previous response was invalid: {error_reason}
Return Thought and Action now:

D Baseline Reproduction

We reproduced Skill1, SkillRise, and Mem0 with Qwen3.5-4B under a shared bank-evaluation protocol. The trained Skill1 and SkillRise checkpoints served as curators for separate bank generation. Each bank contained 50 entries of at most 100 words, constructed from matched bundles of four trajectories. Evaluation used the frozen Qwen3.5-4B executor and `all-mpnet-base-v2` retrieval, with the top 10 entries selected once and held fixed throughout each episode. Skill1 and SkillRise results are reported as means and standard deviations across three runs.

D.1 SKILL1

We retained Skill1's query generation, skill reranking, library updates, and distillation objective (Shi et al., 2026). We trained query-generation and reranking responses while excluding task-execution responses from the policy loss. Complete trajectories still supplied outcome rewards. We used the step-40 checkpoints to generate the evaluation banks.

D.2 SKILLRISE

We retained SkillRise's cross-task training and decoupled credit assignment (Yao et al., 2026). Each trial comprised three tasks from the same family, with the skill document updated between tasks and reset between trials. Solving responses received feedback from their own task outcomes, whereas curation responses received feedback from subsequent tasks.

D.3 MEM0

We evaluated both semantic and procedural banks constructed with Mem0 v2.1.0 (Chhikara et al., 2025) and Qwen3.5-4B, retaining the package's native memory-writing prompts and exporting memories from the same trajectory bundles into the shared bank format. For each evaluation seed, we averaged the results of the two bank types, then reported the mean and standard deviation across three seeds.

D.4 HUMAN KNOWLEDGE CONSTRUCTION

For the Human-written baseline, we sampled trajectories using the same procedure and distributed them to volunteers, who wrote knowledge entries based on these trajectories. We then assembled these entries into knowledge banks following the shared construction protocol and evaluated the resulting banks under the same evaluation protocol.

E EXPERIMENTAL DETAILS

E.1 HYPERPARAMETER DETAILS

Table 4 summarizes the training configuration for the Qwen3.5-4B curator. We update all curator parameters while keeping the executor and reference model frozen. After training, the curator generates separate knowledge banks for downstream evaluation.

E.2 DATA SOURCES AND TASK SPLITS

Table 5 summarizes the task sources used by Knowledge Weaver. The full training split is eligible for reward sampling; each group samples eight tasks rather than evaluating the entire split at every update.

Training trajectories, reward tasks, and bank-generation sources may share training tasks. For ALFWorld, these sources belong to `train`, whereas final evaluation uses `valid_unseen`; the latter was also used for training-time monitoring. For WebShop, official human training and test tasks have disjoint task IDs. Synthetic bank-generation tasks use a separate ID namespace.

E.3 TRAINING DETAILS

We initialized the curator, executor, and reference policy from Qwen3.5-4B. Training updated all curator parameters, while the executor and reference policy remained frozen. We optimized the curator with the clipped GRPO objective using AdamW, a constant learning rate, and KL regularization against the reference policy. Table 4 specifies the principal settings for the main Knowledge Weaver configuration on ALFWorld and WebShop.

The training batch size and GRPO group size refer to different sampling levels. Each training batch contained 32 trajectory contexts, each comprising four trajectories from the same task type. For each context, the curator sampled eight candidate knowledge entries from the identical input, giving

Hyperparameter	ALFWorld	WebShop
Optimization		
Optimizer	AdamW	
Learning rate	1×10^{-6}	
Learning-rate schedule / warmup	Constant / none	
AdamW (β_1, β_2)	$(0.9, 0.999)$	
AdamW numerical epsilon	10^{-8}	
Weight decay	0.01	
Gradient-norm clipping	1.0	
Policy-ratio clipping parameter	0.2	
KL regularization coefficient	0.01	
PPO minibatches per training batch	32	
Context Sampling and Generation		
Trajectory contexts per training batch	32	
Input trajectories per context	4	
Candidate entries per context (GRPO group size)	8	
Curator sampling temperature	1.0	
Curator top-p	1.0	
Maximum generated tokens per candidate	256	
Maximum words per knowledge entry	100	
Maximum rendered steps per source trajectory	24	
Source-observation character limit	120	400
Execution Feedback and Bank Updates		
Executor temperature	0	
Maximum actions per reward rollout	50	15
Reward-task rollouts per candidate and condition	8	
Standalone-success weight, λ_{SR}	0.5	
Marginal-success weight, λ_{MSR}	0.5	
Token-level MI weight, λ_{MI}	0.1	
MI credit cap after scaling, c_{MI}	0.15	
Valid-format bonus / invalid-format penalty	$+0.02$ / -0.1	
Retrieved entries for marginal feedback, m	3	

Table 4: Key training hyperparameters for the main Qwen3.5-4B Knowledge Weaver configuration. Shared values span both benchmark columns; benchmark-specific values are shown separately. Each training batch contains 32 trajectory contexts and eight candidates per context, for 256 candidate entries before optimization. Only the curator is trained. The table describes training; the separately generated evaluation banks contain 50 entries and use $k = 10$ in the main comparison.

Data role	ALFWorld	WebShop
Training source trajectories	Collected from `train` across all six task types.	Collected from the full official human training split.
SR/MSR reward tasks	Sampled from all `train` tasks matching the source context's task type.	Sampled from all 10,587 official human training tasks.
Evaluation-bank source trajectories	120 `train` trajectories, with 20 per task type.	300 synthetic training-task trajectories.
Final evaluation tasks	All 134 `valid_unseen` tasks.	A fixed subset of 100 tasks from the 500 official human test tasks.

Table 5: Task sources for training, bank construction, and final evaluation.

$32 \times 8 = 256$ candidates per batch. Execution rewards were normalized within each group of eight candidates. The value 32 therefore counts input contexts per training batch, whereas the value 8 counts candidate entries per context. The PPO minibatch count in the table specifies how this generated batch was partitioned for optimization.

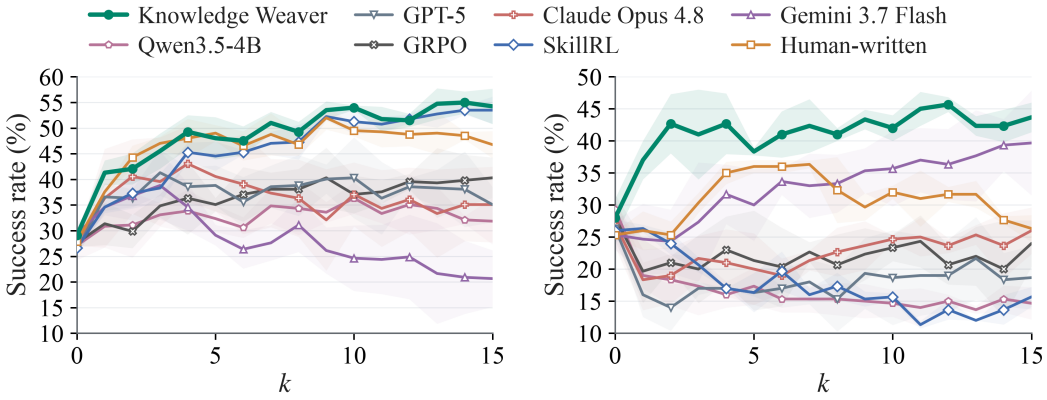
For execution feedback, we used eight reward-task rollouts per candidate and condition, with matched tasks shared across the standalone, bank-only, and bank-plus-candidate conditions. Standalone success and marginal success received equal weights of 0.5. A valid `<skill>` response received a formatting bonus of 0.02; an invalid response received a penalty of -0.1 without execution or MI credit. Token-level MI credit was applied only to knowledge-body tokens, with weight 0.1 and a cap of 0.15 after scaling. We used within-group normalization for execution rewards and batch-level normalization for the resulting token advantages.

The training bank was initialized empty and held fixed while each batch was scored. Marginal feedback used up to three retrieved entries, and bank admission required a strictly positive marginal success gain. On ALFWorld, the update quota allowed at most one entry from each of two task types per batch during the first 20 updates, and at most one entry thereafter. On WebShop, at most one entry was admitted per batch. After training, the curator generated a separate evaluation bank of 50 entries. The training retrieval count of three is distinct from the main evaluation setting of $k = 10$.

E.4 Retrieval-Count Scaling

We evaluate retrieval counts $k \in \{0, \ldots, 15\}$ while keeping each knowledge bank, the executor, and the benchmark task set fixed. For methods also shown in Figure 3, we use the same evaluation runs and replace token count with retrieval count on the horizontal axis.

Each curve uses a fixed set of runs across all retrieval counts using three runs. Each curve retains its own no-knowledge control at $k = 0$. WebShop SkillRise is omitted because its complete retrieval sweep is unavailable.



(a) ALFWorld (b) WebShop

Figure 4: Success rate as the number of retrieved knowledge entries k varies from 0 to 15. The horizontal axis reports the number of retrieved knowledge injected into the frozen executor, averaged over tasks and runs. Lines and shading show the mean and one standard deviation across three runs. Each curve uses its own no-knowledge control at $k = 0$.

Figure 4 shows that the effect of additional retrieved knowledge varies across banks and environments. Knowledge Weaver achieves SRs of 54.0% and 42.0% at $k = 10$ on ALFWorld and WebShop, respectively. At $k = 15$, the corresponding rates are 54.2% and 43.7%.

F Training Algorithm

Algorithm 1 summarizes Knowledge Weaver training. Candidates generated from the same trajectory context form a GRPO sampling group. Token-level MI feedback compares each candidate across source contexts while preserving its generated tokens and prefix. The executor and reference policy remain frozen throughout training.

All execution conditions for a candidate use the same tasks, with B_q fixed in the marginal comparison. With an empty bank, the bank-only condition reduces to execution without knowledge. Invalid

Algorithm 1 Knowledge Weaver curator training

Require: Curator π_θ, frozen executor, reference policy π_{ref}, trajectory pool $\mathcal{T}$
Require: Updates J, candidates per context $G \geq 2$, retrieval count m, admission schedule $\mathcal{Q}$
Require: Reward weights $\lambda_{\text{SR}}, \lambda_{\text{MSR}}$, MI weight λ_{MI}, MI cap c_{MI}
Ensure: Trained curator π_θ

Initialize training knowledge bank $\mathcal{B} \leftarrow \varnothing$
for $j = 1, \ldots, J$ **do**
  $\pi_{\text{old}} \leftarrow \pi_\theta$; fix bank snapshot $\mathcal{B}^{\text{snap}} \leftarrow \mathcal{B}$
  Sample a batch of trajectory-context sets from $\mathcal{T}$
  for each context set $\mathcal{X} = (x_1, \ldots, x_N)$, with $N \geq 2$ **do**
    for $i = 1, \ldots, N$ **do**
      Sample G candidates $z \sim \pi_{\text{old}}(\cdot \mid x_i)$
      Select training reward tasks $\mathcal{D}_i$, shared by these candidates
      Retrieve up to m entries B_q from $\mathcal{B}^{\text{snap}}$ for each $q \in \mathcal{D}_i$
      for each candidate z generated from x_i **do**
        Initialize $U(\{z\}) \leftarrow 0$, $\text{MSR}(z) \leftarrow 0$, and $r_t^{\text{MI}} \leftarrow 0$
        if z has valid formatting **then**
          Evaluate $s(q, \{z\})$, $s(q, B_q)$, and $s(q, B_q \cup \{z\})$ on $\mathcal{D}_i$
          Average standalone success to obtain $U(\{z\})$
          Compute $\text{MSR}(z)$ using Equation 5 on $\mathcal{D}_i$
          Rescore z_t under each $x_\ell \in \mathcal{X}$ with $z_{<t}$ fixed
          Compute masked token credits r_t^{MI} using Equation 6
        end if
        $R_{\text{exec}}(z) \leftarrow \lambda_{\text{SR}} U(\{z\}) + \lambda_{\text{MSR}} \text{MSR}(z) + r_{\text{fmt}}(z)$
      end for
      Normalize $R_{\text{exec}}(z)$ within the G candidates generated from x_i
    end for
  end for
  Combine normalized execution rewards and local MI credit as $A_{g,t}$ (Equation 7)
  Compute $\widehat{A}_{g,t}$ by reference regularization and batch normalization (Equation 16)
  Update θ using the clipped GRPO objective with π_{old} and $\widehat{A}_{g,t}$
  Select valid candidates absent from $\mathcal{B}^{\text{snap}}$ with $\text{MSR}(z) > 0$
  Admit the highest-gain eligible candidates subject to $\mathcal{Q}_j$ to obtain $\mathcal{B}$
end for
return π_θ

candidates receive only the formatting penalty. MI credit is added after execution advantages are computed, so it is not propagated to preceding tokens through return estimation.

The admission schedule $\mathcal{Q}_j$ specifies the eligible task types and update quota. On ALFWorld, we admit at most one candidate per scheduled task type, selecting two types per update for the first 20 updates and one thereafter. On WebShop, we admit at most one candidate per update. Admission ranks candidates by MSR alone; if no eligible candidate has positive gain, the bank remains unchanged. The updated bank is used in the next training batch. After training, the curator generates a separate knowledge bank for downstream evaluation.

F.1 Optimization details

Feedback and final advantages. The combined SR, MSR, and formatting rewards in Equation 7 are normalized across candidates from the same source context before being placed at the final generated token. We use zero value estimates, unit discount, and unit GAE parameter. Reference-policy regularization contributes a penalty at each generated token. Let $\ell_{g,t}^{\text{beh}}$ and $\ell_{g,t}^{\text{ref}}$ denote the sampled token log-probabilities under the behavior and frozen reference policies. The sampled KL term is

$$\kappa_{g,t} = \text{clip}\big(\ell_{g,t}^{\text{beh}} - \ell_{g,t}^{\text{ref}}, -10, 10\big). \tag{15}$$

Backward return computation accumulates these penalties from token t to the entry's final token T_g. Local MI credit is added after this computation, so it is not accumulated across token positions.

Using the modified advantage $A_{g,t}$ from Equation 7, the final advantages are

$$\widehat{A}_{g,t} = \mathrm{Norm}_{\mathrm{batch}}\left(A_{g,t} - \beta \sum_{s=t}^{T_g} \kappa_{g,s}\right). \tag{16}$$

The sum includes only unmasked generated tokens. Batch normalization subtracts the mean and divides by the sample standard deviation plus ϵ_{num} over all such tokens in the rollout batch. Group reward normalization also uses sample standard deviation, with $\epsilon_{\mathrm{num}} = 10^{-5}$ in both operations. The configurations use $\beta = 0.01$ and clipping radius $\varepsilon = 0.2$. Reference regularization enters through these rewards, without an additional differentiated KL term in the loss.

The full configuration uses $\lambda_{\mathrm{SR}} = \lambda_{\mathrm{MSR}} = 0.5$, $\lambda_{\mathrm{MI}} = 0.1$, and $c_{\mathrm{MI}} = 0.15$. Formatting adds 0.02 for valid entries and gives invalid entries a reward of -0.1. The backend also clips scalar rewards before group normalization and local token credits before addition to $[-20, 20]$. This bound is inactive for these reward ranges and does not alter the equations above.

Clipped update. We use the clipped policy surrogate underlying GRPO (Shao et al., 2024), averaged over valid tokens in each optimization minibatch. For token advantages $A = \{A_{g,t}\}$, the loss is

$$\mathcal{L}_{\mathrm{clip}}(\theta; A) = -\frac{1}{|\mathcal{V}|} \sum_{(g,t)\in\mathcal{V}} \min\{\widetilde{\rho}_{g,t}(\theta) A_{g,t},\ \mathrm{clip}(\widetilde{\rho}_{g,t}(\theta), 1-\varepsilon, 1+\varepsilon)\, A_{g,t}\}, \tag{17}$$

where $\mathcal{V}$ contains unmasked token positions in an optimization minibatch and g indexes candidates across source contexts. Training uses $A = \widehat{A}$ from Equation 16, with all advantage quantities held fixed during differentiation. The ratio $\widetilde{\rho}_{g,t}$ compares the current token probability with a detached proximal probability,

$$\widetilde{\rho}_{g,t}(\theta) = \exp\Big(\log \pi_\theta(z_t^{(g)} \mid x^{(g)}, z_{<t}^{(g)}) - \ell_{g,t}^{\mathrm{prox}}\Big), \tag{18}$$

where $x^{(g)}$ is candidate g's source context. Standard GRPO uses the behavior log-probability as $\ell_{g,t}^{\mathrm{prox}}$. Our backend uses a log-linear approximation based on the policy versions recorded during generation. For current training version v and token generation version $v_{g,t}^{\mathrm{beh}} < v$, define

$$\begin{aligned} \alpha_{g,t} &= \mathrm{clip}\left(\frac{v - 1 - v_{g,t}^{\mathrm{beh}}}{v - v_{g,t}^{\mathrm{beh}}}, 0, 1\right), \\ \ell_{g,t}^{\mathrm{prox}} &= (1-\alpha_{g,t})\ell_{g,t}^{\mathrm{beh}} + \alpha_{g,t}\,\mathrm{stopgrad}\Big(\log \pi_\theta(z_t^{(g)} \mid x^{(g)}, z_{<t}^{(g)})\Big). \end{aligned} \tag{19}$$

Otherwise $\alpha_{g,t} = 0$, and the behavior log-probability is used directly. Samples from the immediately preceding version also have $\alpha_{g,t} = 0$. The denominator is detached even when its numerical value incorporates current-policy log-probabilities. We use neither dual clipping nor rejection sampling. Algorithm 1 omits this version bookkeeping for readability; it does not change the knowledge rewards or bank-admission criterion. All updates apply only to the curator, with the executor and reference policy frozen.