SAPO: Step-Level Skill-Augmented Policy Optimization for Multi-Turn LLM Agents

Prince Zizhuang Wang
Carnegie Mellon University
princewang@cmu.edu

Shuli Jiang
Carnegie Mellon University
shulij@alumni.cmu.edu

Abstract

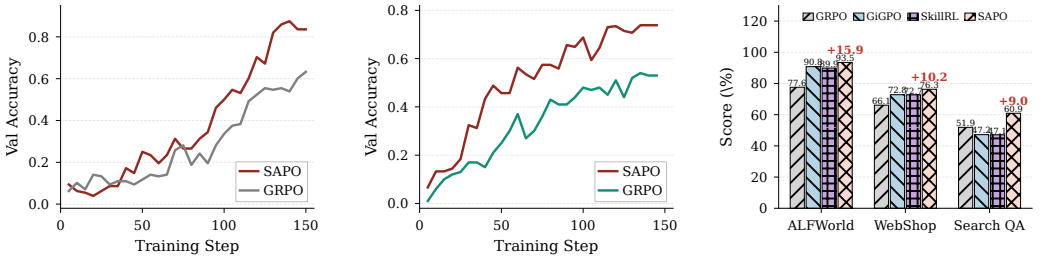
Large Language Model (LLM) agents have shown strong results on multi-turn tool-use tasks, yet they typically operate in isolation during training, failing to leverage skills accumulated across episodes. Existing experience-augmented methods address this by organizing past trajectories into retrievable libraries, but they retrieve skills only once based on the initial task description and hold them constant throughout the episode. In multi-turn settings where observations change at every step, this static retrieval becomes increasingly mismatched as episodes progress. We propose SAPO (Step-Level Skill-Augmented Policy Optimization), a reinforcement-learning framework that retrieves relevant skills at each decision step conditioned on the current observation. SAPO operates through three components: (i) step-level observation clustering that groups structurally equivalent environmental states for efficient cluster-indexed retrieval; (ii) a self-evolving skill bank that distills successful strategies and failure patterns through score-based admission and rate-limited extraction; and (iii) policy optimization with step-level credit assignment for fine-grained advantage estimation across multi-turn episodes. The skill bank evolves alongside the policy through semantic analysis rather than gradient updates. On long-horizon multi-turn agent benchmarks (ALFWorld, WebShop, and seven search-augmented QA tasks), SAPO achieves 93.5% on ALFWorld, 76.3% success on WebShop, and 60.9% average across QA tasks, outperforming both standard RL and prior skill- and experience-augmented baselines. Our code is available at https://github.com/kingofspace0wzz/slea-rl.

1 Introduction

Large language model (LLM) agents [Yao et al., 2022b, Shinn et al., 2023] have demonstrated remarkable capabilities across complex tasks such as web navigation [Yao et al., 2022], embodied planning [Shridhar et al., 2021], and tool-integrated reasoning [Jin et al., 2025]. Reinforcement learning (RL) has become a key paradigm for training these agents, with group-based algorithms such as GRPO [Guo et al., 2025] and Dr. GRPO [Liu et al., 2025] proving especially effective. Despite these advances, each task execution remains largely episodic: current LLM agents operate in isolation, unable to learn from past successes or failures across episodes, which significantly hinders their evolution.

Multi-turn tool-use tasks—such as ALFWorld [Shridhar et al., 2021], WebShop [Yao et al., 2022], and interactive web search—present a fundamentally different challenge than single-turn reasoning [Achiam et al., 2023, Hui et al., 2024, Team et al., 2023, Liu et al., 2024]. Unlike static settings where the context remains fixed throughout generation, multi-turn environments change at every step: each action reveals a new room configuration, each click surfaces new listings. The agent must continuously adapt its reasoning to an ever-changing world state across dozens of sequential decisions. Standard RL approaches [Schulman et al., 2017, Ahmadian et al., 2024, Guo et al., 2025,

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) ALFWorld training curves. (b) WebShop training curves. (c) Method comparison.

Figure 1: (a, b) Validation success rates on ALFWorld and WebShop during RL training (Qwen2.5-1.5B-Instruct): SAPO converges faster and reaches higher asymptotic performance than GRPO. (c) Final scores on ALFWorld, WebShop, and search-augmented QA (avg. across the seven QA tasks in Table 2) with Qwen2.5-7B-Instruct: SAPO outperforms GRPO, GiGPO, and SkillRL on all three; red annotations show SAPO's gain over GRPO.

Zheng et al., 2025a, Feng et al., 2025] face three compounding challenges in this setting. First, each training rollout acts in isolation: episodes begin from scratch with no memory of prior interactions, causing the agent to repeat mistakes and discard successful strategies across thousands of rollouts. Second, most multi-turn environments provide only *sparse, outcome-level rewards*, leaving dozens of intermediate steps without learning signal and making credit assignment difficult. Third, RL training proceeds purely through *weight updates*, discarding the structured, interpretable knowledge that accumulates across episodes in a form that could directly guide future behavior.

Recent work on experience-augmented agents [Xia et al., 2026, Zhang et al., 2026, Cai et al., 2025b] addresses the first limitation by organizing successful trajectories into retrievable libraries. However, these methods perform *task-level* retrieval: a fixed set of skills is fetched once based on the initial task description and held constant for the entire episode. In multi-turn settings, this provides increasingly stale context as the environment state diverges from the original task prompt. Compounding this, episode-level outcome rewards provide no signal at intermediate steps, so even when retrieved skills do guide a useful action, the policy receives no fine-grained feedback about which step the credit belongs to. These limitations raise a core question:

> *Can LLM agents learn to leverage step-specific accumulated skills throughout an episode, matching retrieved strategies to the current observation at every decision point—and how should an agent determine which intermediate actions causally contributed to the final outcome when episodes span dozens of steps?*

The key insight is that the relevance of accumulated skills is *state-dependent*: the strategies useful when a WebShop agent first navigates to a product category differ fundamentally from those needed when comparing two specific items at checkout twenty steps later. Just as not all intermediate actions deserve equal credit for the episode outcome, not all steps benefit equally from the same retrieved skills. Step-level retrieval addresses both dimensions simultaneously: by conditioning retrieved skills on the current observation, agents receive targeted support at exactly the steps where it matters, improving the quality of the exploratory actions that receive fine-grained credit. SAPO unifies these two concerns through observation clustering: the same equivalence classes that index skills for retrieval also serve as the grouping structure for per-step advantage normalization, so retrieval and credit assignment respond consistently to the agent's evolving state.

To this end, we propose **SAPO** *(Step-Level Skill-Augmented Policy Optimization)*, a multi-turn RL framework that integrates step-level skill retrieval directly into the training loop. At each decision step, SAPO retrieves relevant skills from a structured skill bank conditioned on the current observation, injecting them via format-preserving augmentation. To enable efficient retrieval at scale, SAPO clusters structurally equivalent observations into equivalence classes, so that skills proven effective at one state immediately benefit all similar states across trajectories. The skill bank self-evolves through quality-controlled semantic analysis of the best and worst trajectories each epoch, with score-based admission and rate-limited extraction maintaining quality under continuous update

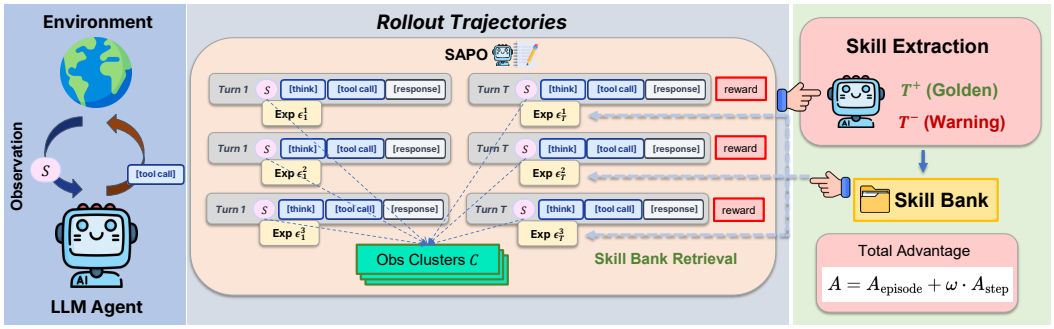


Figure 2: Overview of SAPO compared to standard GRPO. **Left**: the LLM agent interacts with the environment in a multi-turn loop; a unified skill bank $\mathcal{B}$ provides step-level guidance. **Center**: GRPO (top, gray) samples G trajectories without skill augmentation, while SAPO (bottom, orange) retrieves skills $\varepsilon_t^{(i)}$ at each step via cluster-indexed lookup over observation clusters $\mathcal{C}$.

pressure. Crucially, skills are used during both training and inference, ensuring that the policy learns to effectively leverage retrieved skills rather than treating them as a test-time artifact.

Our main contributions are:

- **Observation clustering as a unifying primitive for step-level retrieval and credit.** Unlike existing methods that retrieve skills once per task and reward only at episode end, SAPO retrieves at each decision step conditioned on the current observation, and the same observation-cluster index drives per-step advantage normalization, yielding a multi-level objective $\hat{A} = A_{\text{episode}} + w \cdot A_{\text{step}}$ rather than treating retrieval and credit as decoupled modules.
- **Self-evolving skill bank with dual-source extraction and score-based admission.** SAPO distills skills from *both* successful trajectories (direct strategy distillation) and failed trajectories (reflection-based corrective skills) into a unified bank; score-based admission with rate-limited extraction lets the bank co-evolve with the policy through semantic analysis rather than gradient updates, using only the same base model for skill extraction and requiring no external reflector LLM.
- **Introduce SAPO and empirical validation on long-horizon benchmarks.** We introduce Step-Level Skill-Augmented Policy Optimization (SAPO) for multi-turn agent training and achieve strong results on ALFWorld, WebShop, and seven search-augmented QA tasks, outperforming many standard RL and experience-augmented baselines.

2 Preliminaries

2.1 LLM Agent Framework

We consider an LLM agent operating in an interactive environment $\mathcal{E}$. At each timestep t, the agent observes a state $o_t \in \mathcal{O}$, selects an action $a_t \in \mathcal{A}$, and receives a reward r_t and next observation o_{t+1}. A trajectory $\tau = (o_0, a_0, r_0, \ldots, o_T, a_T, r_T)$ captures one episode of interaction. Tasks are specified by natural language descriptions d. An LLM-based agent parameterized by θ implements a policy $\pi_\theta(a_t \mid o_{\leq t}, d, c)$, where c represents additional context (e.g., retrieved skills). Our goal is to learn a policy that maximizes expected return $\max_\theta \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$ subject to context length constraints $|c| \leq L_{\max}$. In many environments, the agent receives only a sparse terminal reward $R(\tau) \in \{0, 1\}$ indicating task success or failure, with intermediate rewards $r_t = 0$ for all $t < T$. In this setting, the discounted return from step t reduces to $\hat{R}_t = \gamma^{T-t} R(\tau)$. Our framework accommodates both sparse and dense reward settings; we use r_t in full generality and specialize where needed. This sparse reward structure makes step-level credit assignment particularly challenging, as the agent must determine which intermediate actions causally contributed to the final outcome across episodes spanning dozens of steps.

2.2 Group Relative Policy Optimization

Group Relative Policy Optimization (GRPO) [Guo et al., 2025] is a reinforcement learning method that avoids training a critic by using intra-group relative rewards to optimize the policy. For each query x, the model samples G responses $\{y^{(1)}, \ldots, y^{(G)}\}$, which are scored to obtain rewards $\{R^{(1)}, \ldots, R^{(G)}\}$, where $i \in [G]$ indexes the trajectory within the sampled group. GRPO computes normalized advantages and updates the policy with a PPO-style clipped objective [Schulman et al., 2017]:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{x,\{y^{(i)}\}} \left[\frac{1}{G} \sum_{i=1}^{G} \min \left(\rho^{(i)} A^{(i)}, \text{clip}(\rho^{(i)}, 1-\epsilon, 1+\epsilon) A^{(i)} \right) - \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) \right], \quad (1)$$

where $\rho^{(i)} = \frac{\pi_\theta(y^{(i)}|x)}{\pi_{\text{old}}(y^{(i)}|x)}$ is the importance ratio and $A^{(i)} = \frac{R^{(i)} - \text{mean}(\{R^{(j)}\}_{j=1}^{G})}{\text{std}(\{R^{(j)}\}_{j=1}^{G})}$ is the normalized advantage, with ϵ and β as hyperparameters, π_{old} the policy before the current update, and π_{ref} the reference policy for KL regularization. This formulation eliminates value-network training while providing automatic baseline subtraction and variance normalization. A critical property for time efficiency is that sampling G trajectories in parallel using modern inference engines incurs wall-clock time similar to sampling a single trajectory, making group-based methods highly practical.

3 SAPO : Step-level Skill-Augmented Policy Optimization

We propose SAPO, a multi-turn reinforcement learning framework that integrates step-level skill retrieval into agent training. The framework contains three interleaved components: (i) a self-evolving skill bank, which distills reusable skills from successful and failed trajectories through quality-controlled semantic analysis; (ii) step-level observation clustering with cluster-indexed retrieval, enabling efficient retrieval of relevant past skills; and (iii) skill-augmented rollouts with policy optimization, where retrieved skills are incorporated into the agent context during training to guide decision-making. Figure 2 summarizes the framework.

3.1 Skill Bank

The skill bank $\mathcal{B}$ is a structured repository of textual skills accumulated during training. Each entry $e \in \mathcal{B}$ is represented as a tuple $e = (s_e, l_e, z_e)$, where s_e is a natural-language description of an actionable skill, $l_e \in \{\text{principle}, \text{pattern}, \text{example}\}$ denotes its abstraction level, and $z_e \in \mathbb{R}$ is a quality score derived from the reward of the trajectory from which the skill is extracted. Skills capture both successful strategies (e.g., effective decision sequences and reasoning patterns) and corrective guidance distilled from failures (e.g., instructions of the form "in state X, do Y"), all written as positive, actionable advice. To control growth, the bank capacity is limited to C entries per abstraction level. We use $\varepsilon_t \subset \mathcal{B}$ to denote the subset of skills retrieved at step t for a given observation. The bank evolves through semantic analysis rather than gradient updates (Section 3.4), providing interpretable knowledge structures that can be inspected and transferred across agents.

3.2 Step-Level Observation Clustering

We introduce an observation clustering mechanism to enable efficient skill retrieval and consistent step-level credit assignment. In multi-turn environments, many observations are semantically similar; clustering allows such states to share both advantages and retrieved skills. Prior work groups similar observations for advantage normalization [Feng et al., 2025]; SAPO extends this idea by using the same structure to index reusable skills.

We maintain a cluster index $\mathcal{C} = \{c_1, \ldots, c_M\}$, where each cluster c_i is represented by a prototype observation $\text{rep}(c_i)$ and stores an associated skill pool $\mathcal{B}_{c_i} \subseteq \mathcal{B}$. Given a new observation o_t, we assign it to an existing cluster or create a new one:

$$c(o_t) = \begin{cases} c_i & \text{if } \exists\, c_i : \ \text{sim}(o_t, \text{rep}(c_i)) \geq \delta, \\ \text{new cluster} & \text{otherwise,} \end{cases} \quad (2)$$

where $\text{sim}(\cdot, \cdot)$ is a sequence similarity function and δ is the similarity threshold. If the assigned cluster already contains skills, retrieval is performed directly from its pool:

$$\varepsilon_t = \text{TopK}\big(\mathcal{B}_{c(o_t)}, k\big), \quad (3)$$

where k controls the number of retrieved skills. This cluster-indexed retrieval avoids scanning the full bank. If the cluster has no associated skills, the method falls back to bank-wide semantic retrieval:

$$\varepsilon_t = \mathrm{TopK}(\{e \in \mathcal{B} : \mathrm{sim}(o_t, e) > \delta\}, k)\,, \tag{4}$$

and the retrieved skills are linked to the cluster for future reuse. As training progresses, most observations match existing clusters, making retrieval increasingly efficient and targeted. During bank evolution (Section 3.4), newly admitted skills are associated with relevant clusters, while score-based admission replaces lower-quality entries, allowing cluster pools to improve alongside the policy.

3.3 Skill-Augmented Rollout

Building on the observation clustering, SAPO performs *step-level skill retrieval* during rollout. At step t in a trajectory, the agent observes o_t and retrieves relevant skills ε_t using the cluster-indexed mechanism (Eqs. 3–4). To avoid injecting low-quality skills early in training, retrieval is activated only after a warmup of W epochs and when the bank size exceeds $|\mathcal{B}| > C_{\min}$. When retrieval is enabled, the policy conditions on both the current observation and the retrieved skills:

$$a_t \sim \pi_\theta(\cdot \mid \mathrm{augment}(o_t, \varepsilon_t))\,, \tag{5}$$

where skills are inserted into the prompt via format-preserving augmentation (Appendix A). The key distinction from prior work [Zhang et al., 2025, Cai et al., 2025b] is that skill retrieval occurs *at every step* based on the current observation, rather than once from the initial task description. This design allows skill guidance to adapt to evolving agent states and ensures that the model learns to utilize retrieved skills during training under the same conditions used at inference.

3.4 Self-Evolving Skill Bank

The skill bank evolves through *quality-controlled semantic analysis* rather than gradient updates. Naively extracting skills from all trajectories would generate hundreds of candidates per step, causing eviction churn and unstable retrieval. SAPO addresses this with three controls: outcome partitioning, selective extraction, and score-based admission.

Outcome partitioning. After rollout, trajectories are split by reward. Let $r_j = R(\tau_j)$ denote the reward of trajectory τ_j. Using a threshold η (typically the batch median), we form

$$\mathcal{T}^+ = \{\tau_j : r_j > \eta\}, \qquad \mathcal{T}^- = \{\tau_j : r_j \le \eta\}, \tag{6}$$

so that successes and failures from the same batch can be contrasted in a single extraction call (below).

Contrastive extraction. An auxiliary LLM (the same base model used for rollout) receives the top-K_{traj} successes from $\mathcal{T}^+$ together with the bottom-K_{traj} failures from $\mathcal{T}^-$ in a single call and synthesizes forward-looking skills by contrasting the two: it identifies the decisions that distinguished successful from failed trajectories and emits up to K_{skill} actionable skills, each phrased as a positive instruction ("in state X, do Y") rather than a prohibition. Each skill inherits the episode reward of its anchoring successful trajectory as its quality score z_e. This contrastive form makes failures useful without storing prohibitions: a failed trajectory contributes a skill that says what should have been done, drawing on a successful peer for the answer.

Score-based admission. The bank has capacity C per abstraction level. A candidate skill is admitted only if its quality score z_e exceeds the lowest-scoring entry at the same level; otherwise it is discarded. As the policy improves, higher-reward trajectories yield higher-quality skills that replace older ones, keeping the bank aligned with the agent's evolving capabilities. Accepted skills are inserted into $\mathcal{B}$ and associated with relevant observation clusters.

3.5 Policy Optimization

SAPO combines episode-level and step-level credit assignment. For each task, we sample G trajectories with skill-augmented rollout and compute episode-level advantages following GRPO. To provide fine-grained credit, we introduce *step-level advantages* based on observation clusters (Section 3.2). For each timestep t across all G trajectories, we assign the observation $o_t^{(i)}$ to its nearest cluster c and collect all (action, discounted return) pairs from steps assigned to c into $\mathcal{S}(c) = \{(a_t^{(i)}, \hat{R}_t^{(i)}) \mid c(o_t^{(i)}) = c\}$, where $\hat{R}_t^{(i)} = \sum_{k=t}^{T} \gamma^{k-t} r_k^{(i)}$ is the discounted return from

per-step environment rewards (distinct from the episode reward $R(\tau^{(i)})$). Within each such group, we normalize the discounted returns to obtain per-step advantages

$$A_{\text{step}}(a_t^{(i)}) = \frac{\hat{R}_t^{(i)} - \text{mean}\Big(\{\hat{R}_t^{(j)} \mid (a_t^{(j)}, \hat{R}_t^{(j)}) \in \mathcal{S}(c)\}\Big)}{\text{std}\Big(\{\hat{R}_t^{(j)} \mid (a_t^{(j)}, \hat{R}_t^{(j)}) \in \mathcal{S}(c)\}\Big)}, \tag{7}$$

so actions taken at the same observation cluster but leading to different outcomes receive different credit. The combined advantage is $\hat{A}(a_t^{(i)}) = A_{\text{episode}}(\tau^{(i)}) + w \cdot A_{\text{step}}(a_t^{(i)})$, where w controls the relative weight of step-level credit. The policy is updated using the clipped surrogate objective extended with the multi-level advantage:

$$\mathcal{J}(\theta) = \mathbb{E}_{d,\{\tau^{(i)}\}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{t=1}^{T_i}\min\Big(\rho_t^{(i)}\hat{A}_t^{(i)},\ \text{clip}(\rho_t^{(i)}, 1-\epsilon, 1+\epsilon)\,\hat{A}_t^{(i)}\Big) - \beta\, D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})\right], \tag{8}$$

where $\rho_t^{(i)} = \pi_\theta(a_t^{(i)} \mid o_{\le t}^{(i)}, d, \varepsilon_t)/\pi_{\text{old}}(a_t^{(i)} \mid o_{\le t}^{(i)}, d, \varepsilon_t)$ is the per-step importance ratio conditioned on the skill-augmented context. Importantly, SAPO requires no additional loss terms or auxiliary passes: skill-guided learning arises directly from rollout, where retrieved skills influence actions and are reinforced through their impact on returns.

4 Experiments

We present empirical evaluations of SAPO across a variety of agentic tasks. Our experiments aim to answer the following questions: **(1)** How does SAPO compare to state-of-the-art RL and experience-augmented baselines? **(2)** What is the contribution of each component (step-level retrieval, observation clustering, self-evolving skill bank)? **(3)** How does step-level retrieval compare to task-level retrieval? **(4)** How does the skill bank scale with training?

4.1 Experimental Setup

Benchmarks. We evaluate on two multi-turn agent benchmarks, *ALFWorld* [Shridhar et al., 2021] and *WebShop* [Yao et al., 2022], as well as search-augmented QA tasks covering both single-hop and multi-hop settings (Appendix C).

Baselines. For ALFWorld and WebShop, we compare against: (a) *closed-source LLMs* (GPT-4o, Gemini-2.5-Pro); (b) *prompt-based and memory methods* that guide behavior via in-context reasoning or external memory without parameter updates (ReAct, Reflexion, Mem0, ExpeL, MemP, SimpleMem); (c) *RL methods* with trajectory-level or group-based advantage estimation (RLOO, GRPO, GiGPO, IGPO, HCAPO [Tan et al., 2026]); and (d) *experience-augmented RL* that integrates persistent memory into the RL loop (MemRL, Mem0+GRPO, SimpleMem+GRPO, SkillRL). For search-augmented QA, we compare against R1-Instruct, Search-R1, ZeroSearch, StepSearch, EvolveR, and RL baselines PPO, Reinforce++, GSPO, and GiGPO.

Implementation details. We use Qwen2.5-7B-Instruct as the primary base model, and additionally evaluate Qwen2.5-1.5B-Instruct to study scaling. RL training uses a learning rate of 1×10^{-6}, batch size 16, group size G=8, and 4 gradient accumulation steps, with clipping ϵ=0.2 and KL coefficient β=0.01. The step-level advantage weight is w=1.0. For the skill bank, we set the similarity threshold δ=0.85, warmup W=5 epochs, and retrieve top-3 skills per step. Skill extraction and semantic analysis are performed by the same base model used for rollout, requiring no additional model deployment. Per-step retrieval adds minimal overhead: cluster-indexed lookup uses string similarity ($O(|\mathcal{C}|)$ comparisons) rather than neural embeddings, and the retrieved text is prepended to the existing prompt without additional forward passes. The main computational addition is the auxiliary LLM call for skill extraction, which runs once per training batch. For search-augmented QA, we follow the same setup as Search-R1, using E5 as the retriever with group size G=5. Full details are provided in Appendix D.

4.2 Main Results

ALFWorld and WebShop. Table 1 reports results on ALFWorld and WebShop alongside two concurrent baselines: EMPG [Wang et al., 2025b] (action-level intrinsic reward via dynamic entropy)

Table 1: Performance on ALFWorld and WebShop. We report average success rate (%) for ALFWorld subtasks and overall, and both average score and success rate (%) for WebShop. *: results from Feng et al. [2025]. Best and second-best per column are highlighted with red and blue backgrounds. The full table including the prompt-based models is included in Appendix J.

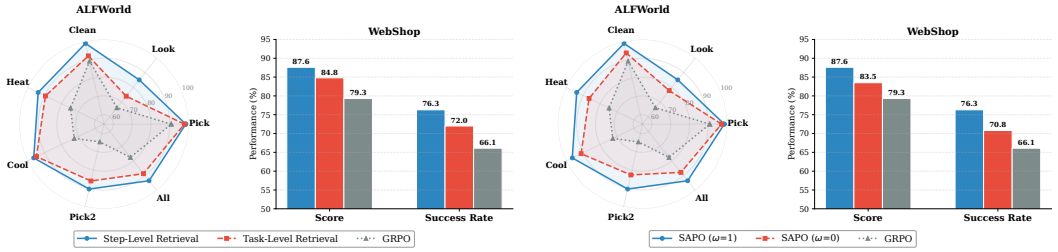
		ALFWorld							WebShop	
Type	**Method**	Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Succ.
Qwen2.5-7B-Instruct										
	RLOO*	87.6	78.2	87.3	81.3	71.9	48.9	75.5	80.3	65.7
	GRPO*	90.8	66.1	89.3	74.7	72.5	64.7	77.6	79.3	66.1
	GiGPO*	97.7	82.7	98.8	83.7	89.3	79.2	90.8	84.4	72.8
	EMPG	92.9	75.2	74.8	86.3	73.7	65.3	78.5	81.0	69.3
	HCAPO	99.1	90.3	97.3	81.8	90.8	81.9	91.4	85.1	73.8
	MemRL	62.8	38.5	22.2	12.5	8.0	0.0	21.4	29.5	9.2
	Mem0+GRPO	78.1	54.8	56.1	31.0	65.0	26.9	54.7	58.1	37.5
	SimpleMem+GRPO	89.5	36.3	60.0	50.0	64.9	26.3	62.5	67.8	46.9
	SkillRL	97.9	71.4	90.0	90.0	95.5	87.5	89.9	85.2	72.7
	SAPO	98.5	85.1	99.0	93.8	96.4	90.4	93.5	87.6	76.3
Qwen2.5-1.5B-Instruct										
	GRPO*	85.3	53.7	84.5	78.2	59.7	53.5	72.8	75.8	56.8
	GiGPO*	94.4	67.5	94.8	94.4	79.8	76.4	86.7	83.1	65.0
	EMPG	85.5	33.5	78.9	76.2	74.7	69.1	73.7	80.4	60.8
	HCAPO	88.6	75.0	97.6	90.7	84.2	74.2	87.0	83.8	68.5
	SAPO	92.6	80.0	96.2	94.7	95.0	76.5	87.5	88.7	75.4

Table 2: Performance on search-augmented QA tasks. SAPO is trained on NQ and HotpotQA. †: in-domain datasets. Best and second-best per column highlighted with red and blue backgrounds.

		Single-Hop QA				Multi-Hop QA			
Type	**Method**	NQ†	TriviaQA	PopQA	HotpotQA†	2Wiki	MuSiQue	Bamboogle	Avg.
Qwen2.5-7B-Instruct									
	RLOO	40.7	72.5	43.1	49.6	55.0	62.2	24.8	49.7
	PPO	38.7	75.4	48.7	48.6	59.7	63.4	26.2	51.5
	GRPO	40.3	77.0	49.6	48.9	57.7	65.1	25.0	51.9
	Reinforce++	34.3	67.5	44.3	45.9	54.5	61.2	23.7	47.3
	GSPO	41.5	77.7	45.4	46.3	60.1	67.6	25.4	52.0
	GiGPO	46.4	64.7	46.1	41.6	43.6	18.9	68.9	47.2
	IGPO	46.7	80.1	52.5	57.2	68.2	31.4	74.9	58.7
	SkillRL	45.9	63.3	73.8	45.9	43.2	20.2	40.3	47.1
	HCAPO	46.1	65.5	47.6	42.1	43.1	17.7	69.0	48.3
	SAPO	48.5	81.8	55.2	59.8	70.5	77.2	33.2	60.9

and HCAPO [Tan et al., 2026] (LLM-as-critic hindsight credit assignment). At Qwen2.5-7B-Instruct, SAPO is best on aggregate ALFWorld success rate, on WebShop score, and on WebShop success rate, and leads the four subtasks that require longer multi-step planning (Clean, Heat, Cool, Pick2). It finishes second to HCAPO on the two short-horizon subtasks (Pick and Look), consistent with HCAPO's hindsight LLM critic giving the sharpest single-step credit. Among skill- and experience-augmented baselines, SAPO improves over SkillRL by 3–4 points on both benchmarks, supporting the claim that step-level retrieval conditioned on the current observation provides more targeted guidance than task-level skill retrieval. At Qwen2.5-1.5B-Instruct, the same pattern holds: SAPO is best on the aggregate ALFWorld column and on WebShop, with the WebShop success-rate gain over GiGPO ($\sim$10 points) consistent with skill augmentation benefiting smaller models with less internalized knowledge.

Search-augmented QA. Table 2 shows results on search-augmented QA. SAPO achieves the best average score (60.9% vs. 58.7% for the strongest baseline IGPO), with the largest gains on the multi-hop tasks requiring iterative information synthesis—most strikingly on MuSiQue, where SAPO improves by $\sim$10 points over GSPO (the best non-SAPO method on this dataset) while the skill- and group-based baselines GiGPO, IGPO, and SkillRL all collapse below 32%. SAPO also generalizes from in-domain training (NQ and HotpotQA) to all four out-of-domain benchmarks. The exception is Bamboogle, where IGPO retains a clear advantage; we attribute this to the dataset's small size and distinctive reasoning patterns that favor IGPO's information-theoretic grouping. Relative



(a) Step-level vs. task-level retrieval.

(b) Step-level advantage (ω=0 vs. ω=1).

Figure 3: Ablation studies on ALFWorld and WebShop (Qwen2.5-7B-Instruct). GRPO is shown as the vanilla RL baseline. (a) Step-level retrieval consistently outperforms task-level retrieval across all subtasks. (b) Removing step-level advantage (ω=0) degrades performance below GRPO on several metrics.

Table 3: Multi-seed comparison with GiGPO on ALFWorld and WebShop: mean $\pm$ standard deviation over three random seeds. *: reported by Feng et al. [2025], taking the stronger of its two advantage-normalization variants for each metric. Higher means in **bold**.

	ALFWorld	WebShop	
Method	All	Score	Succ.
Qwen2.5-1.5B-Instruct			
GiGPO*	86.7 $\pm$1.7	83.5 $\pm$1.8	67.4 $\pm$4.5
SAPO	**88.6** $\pm$2.0	**86.9** $\pm$3.2	**76.1** $\pm$1.2
Qwen2.5-7B-Instruct			
GiGPO*	90.8 $\pm$1.3	86.2 $\pm$2.6	75.2 $\pm$3.8
SAPO	**94.4** $\pm$2.1	**87.4** $\pm$0.8	**76.8** $\pm$1.1

Table 4: Effect of the clustering threshold δ on WebShop with Qwen2.5-1.5B-Instruct. Cluster size is the mean number of observations per cluster.

δ	Mean cluster size	Score	Succ.
0.25	490.5	84.9	61.3
0.85 (default)	193.1	**88.7**	**75.4**
0.95	91.9	61.2	32.0

Table 5: Skill-source comparison on WebShop with Qwen2.5-1.5B-Instruct (δ=0.85). Both variants retrieve three skills per step.

Skill source	Top-3 allocation	Score	Succ.
Success + failure	2 success + 1 failure	**88.7**	**75.4**
Success only	3 success	84.6	70.3

to SkillRL—which excels on PopQA via task-specific retrieval but degrades sharply on multi-hop tasks—SAPO's step-level retrieval delivers more balanced improvements across single-hop and multi-hop settings.

Robustness across seeds. Tables 1 and 2 report one training run per SAPO configuration. To test whether the ALFWorld and WebShop gains hold up under RL training variance, Table 3 compares SAPO over three random seeds with GiGPO, the step-level credit-assignment method that SAPO's policy optimization builds on; GiGPO's statistics are taken from Feng et al. [2025], whose ALFWorld and WebShop protocol SAPO follows. SAPO attains a higher mean on all six metrics. The gains are clearest on WebShop success with Qwen2.5-1.5B-Instruct (76.1 vs. 67.4) and on ALFWorld success with Qwen2.5-7B-Instruct (94.4 vs. 90.8), where the mean $\pm$ std ranges of the two methods do not overlap. On the other four metrics the margins are smaller (1.2–3.4 points) and the ranges overlap, so these gains are consistent in sign but not resolved by three seeds. The single-run SAPO results in Table 1 fall within the corresponding multi-seed ranges.

4.3 Ablation Studies

Task-level vs. step-level retrieval. Figure 3a compares step-level retrieval (ours) with task-level retrieval, where skills are retrieved once from the initial task and kept fixed. Step-level retrieval consistently outperforms task-level retrieval across all ALFWorld subtasks, with the largest gains on Look (+11.1%) and Pick2 (+4.4%), which require adapting to evolving states. On WebShop, it improves score (87.6 vs. 84.8) and success rate (76.3% vs. 72.0%). Both variants significantly outperform GRPO (77.6% on ALFWorld, 66.1% on WebShop), showing that skill augmentation is beneficial, while step-level conditioning provides additional gains. Methodology-wise, the task-level retrieval setting is equivalent to SkillRL.

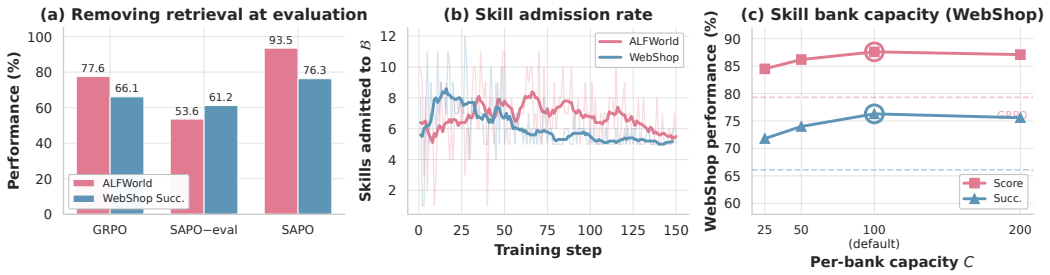


Figure 4: Three views of SAPO's skill bank. **(a)** Removing retrieval at evaluation (training unchanged) drops performance below the no-skill GRPO baseline on both benchmarks, showing that the policy learns to act on retrieved skills as an integrated part of its decision-making. **(b)** Skills admitted to $\mathcal{B}$ per training step on ALFWorld and WebShop; the admission rate decays as the bank fills, consistent with convergence to a stable set of high-utility skills. **(c)** Effect of skill bank capacity C on WebShop; performance peaks at the default C=100 and slightly degrades at C=200. All configurations outperform GRPO (dashed lines).

Step-level advantage. Figure 3b evaluates the impact of step-level credit assignment by comparing w=1 (full method) and w=0 (episode-level only). Removing step-level advantage reduces ALFWorld performance from 93.5% to 87.8%, with larger drops on Look (85.1% $\rightarrow$ 77.8%) and Heat (93.8% $\rightarrow$ 86.5%), and lowers WebShop success from 76.3% to 70.8%. While w=0 still outperforms GRPO, step-level advantage provides additional gains (+5.7% on ALFWorld, +5.5% on WebShop), indicating that fine-grained credit assignment complements skill retrieval.

Cluster-indexed retrieval. We compare cluster-indexed retrieval against full-library top-k retrieval (scanning all $|\mathcal{B}|$ entries) and a random-retrieval sanity check (Appendix K.2). Random retrieval matches the no-skill GRPO baseline on ALFWorld (76.4 vs. 77.6), confirming that arbitrary skill prompts are neutral; full-library retrieval recovers most of the gain (89.4) but trails cluster-indexed (93.5) by 3–4 points across all three metrics. This shows the equivalence-class structure localizes retrieval to genuinely similar states rather than merely reorganizing the same content.

Clustering threshold δ. Because the same observation clusters index skill retrieval and define the groups for step-level advantage normalization, δ controls both how widely skills are shared and which actions are compared, and Table 4 tests the two roles jointly on WebShop with Qwen2.5-1.5B-Instruct. Permissive grouping at δ=0.25 more than doubles the mean cluster size (490.5 vs. 193.1 at the default δ=0.85) and lowers success from 75.4% to 61.3%, yet it still improves over GRPO (75.8 / 56.8%) by 9.1 score and 4.5 success points. Strict grouping at δ=0.95 roughly halves the mean cluster size (91.9) and drops success to 32.0%, well below GRPO, as fragmented clusters limit how far skills can be reused across trajectories. SAPO therefore does not require near-exact observation matches, but it does require enough cross-trajectory sharing. Performance is sensitive to δ, and the default balances sharing against state specificity.

Skill source. By default, the three skills retrieved per step comprise two distilled from successful trajectories and one corrective skill distilled from failed trajectories. To isolate what failure-derived guidance contributes, Table 5 keeps the budget at three skills and fills the failure-derived slot with a third success-derived skill. Success-only retrieval lowers the WebShop score from 88.7 to 84.6 and success from 75.4% to 70.3%. Since the retrieved context stays at three skills, the gap reflects the content of failure-derived guidance rather than additional prompt text: failures supply corrective information that more successful strategies do not recover. This comparison measures the value of each skill source in aggregate; attributing credit to individual skills would require matched rollouts with and without each skill, which we leave to future work.

Skill bank analysis. Figure 4 examines three properties of the unified skill bank. *(a) Reliance on runtime retrieval.* Removing skill retrieval at evaluation while keeping training unchanged drops the policy below the no-skill GRPO baseline on both ALFWorld (53.6 vs. 77.6) and WebShop success rate (61.2 vs. 66.1), and well below the full SAPO (93.5 / 76.3); SAPO's training tightly couples the policy to the retrieved-skill context, and pulling skills out at test time induces a distribution shift (further analyzed in Appendix K.1). *(b) Bank convergence.* The per-step number of skills admitted to $\mathcal{B}$ decays after an initial burn-in on both benchmarks, indicating that the score-based admission

rule converges on a stable set of high-utility skills rather than churning indefinitely (full curves in Appendix K.3). *(c) Capacity sensitivity.* Performance improves from C=25 (84.5 / 71.8%) to C=100 (87.6 / 76.3%), then slightly declines at C=200 (87.1 / 75.6%): overly large banks admit lower-quality skills that reduce retrieval precision. All settings outperform GRPO (79.3 / 66.1%), indicating robustness to C. We use C=100 as the default. A wall-clock training-cost comparison is provided in Appendix K.4.

5 Related Work

Reinforcement learning for LLM agents. RL has become the dominant paradigm for post-training of LLMs [Schulman et al., 2017, Bai et al., 2022, Rafailov et al., 2023], with recent work extending these techniques to multi-turn agentic settings. Group-based methods avoid training a separate critic: RLOO [Ahmadian et al., 2024] uses leave-one-out baselines, GRPO [Guo et al., 2025] computes group-relative advantages, and GSPO [Zheng et al., 2025a] and Reinforce++ [Hu et al., 2025] further refine variance reduction. For multi-turn tasks, GiGPO [Feng et al., 2025] provides step-level credit assignment by grouping similar observations, SPEAR [Qin et al., 2025] employs self-imitation learning, Turn-PPO [Li et al., 2025] applies per-turn optimization, IGPO [Wang et al., 2025a] uses information-theoretic grouping, and HCAPO [Tan et al., 2026] performs hindsight credit assignment for long-horizon LLM agents.

Experience-augmented methods for LLM agents. Retrieval-augmented generation [Lewis et al., 2020] enhances LLMs by conditioning on retrieved content. For agentic tasks, prompt-based methods such as ReAct [Yao et al., 2022b], Reflexion [Shinn et al., 2023], and ExpeL [Zhao et al., 2024] leverage in-context reasoning or external memory without parameter updates, while Mem0 [Chhikara et al., 2025], MemP [Fang et al., 2025], and SimpleMem [Liu et al., 2026] maintain persistent memory stores. Training-free experience-based approaches [Cai et al., 2025a,b] achieve agent adaptation through semantically evolving experience libraries without gradient computation. More recently, methods integrating experience with RL training have emerged: Zhang et al. [2025] introduce experience replay during RL, MemRL [Zhang et al., 2026] couples memory updates with policy optimization, and SkillRL [Xia et al., 2026] retrieves task-level skills to augment rollouts. However, these approaches retrieve skills (or experiences) at the task level—once per episode based on the initial description—which becomes stale as observations evolve across steps. A more comprehensive discussion of baseline comparisons is provided in Appendix G.

6 Conclusion

We propose SAPO, a multi-turn reinforcement learning framework that integrates step-level skill retrieval with quality-controlled skill evolution. By combining a self-evolving skill bank, cluster-indexed retrieval, and multi-level credit assignment, SAPO enables agents to reuse and refine turn-level knowledge throughout training. Experiments across multiple benchmarks demonstrate consistent improvements over strong baselines, with the largest gains appearing on harder, longer-horizon subtasks where the environment state diverges most from the initial task description. These results highlight the importance of aligning skill retrieval with evolving agent states, providing a practical pathway for more efficient and adaptive agent learning. Broader impact and potential risks are discussed in Appendix I.

Acknowledgments and Disclosure of Funding

This work received no third-party funding; all experiments were conducted using the authors' own resources. The authors declare no competing interests.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12248–12267, 2024.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Yuzheng Cai, Siqi Cai, Yuchen Shi, Zihan Xu, Lichao Chen, Yulei Qin, Xiaoyu Tan, Gang Li, Zongyi Li, Haojia Lin, et al. Training-free group relative policy optimization. *arXiv preprint arXiv:2510.08191*, 2025a.

Zhicheng Cai, Xinyuan Guo, Yu Pei, Jiangtao Feng, Jinsong Su, Jiangjie Chen, Ya-Qin Zhang, Wei-Ying Ma, Mingxuan Wang, and Hao Zhou. Flex: Continuous agent evolution via forward learning from experience. *arXiv preprint arXiv:2511.06449*, 2025b.

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.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. *arXiv preprint arXiv:2508.06433*, 2025.

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

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. REINFORCE++: Stabilizing critic-free policy optimization with global advantage normalization. *arXiv preprint arXiv:2501.03262*, 2025.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2.5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, 2017.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

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

Junbo Li, Peng Zhou, Rui Meng, Meet P Vadera, Lihong Li, and Yang Li. Turn-ppo: Turn-level advantage estimation with ppo for improved multi-turn rl in agentic llms. *arXiv preprint arXiv:2512.17008*, 2025.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

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

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, 2023.

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

Yulei Qin, Xiaoyu Tan, Zhengbao He, Gang Li, Haojia Lin, Zongyi Li, Zihan Xu, Yuchen Shi, Siqi Cai, Renting Rui, et al. Learn the ropes, then trust the wins: self-imitation with progressive exploration for agentic reinforcement learning. *arXiv preprint arXiv:2509.22601*, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

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

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, pages 8634–8652, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. URL `https://arxiv.org/abs/2010.03768`.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Fei Huang, and Yan Zhang. Zerosearch: Incentivize the search capability of LLMs without searching. *arXiv preprint arXiv:2505.04588*, 2025.

Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. Hindsight credit assignment for long-horizon LLM agents. *arXiv preprint arXiv:2603.08754*, 2026.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

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

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, and Zhenzhe Ying. Information gain-based policy optimization: A simple and effective approach for multi-turn llm agents. *arXiv preprint arXiv:2510.14967*, 2025a.

Jiawei Wang, Jiacai Liu, Yuqian Fu, Yingru Li, Xintao Wang, Yuan Lin, Yu Yue, Lin Zhang, Yang Wang, and Ke Wang. Harnessing uncertainty: Entropy-modulated policy gradients for long-horizon LLM agents, 2025b. URL https://arxiv.org/abs/2509.09265.

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

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.

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

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, 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 *The Eleventh International Conference on Learning Representations*, 2022b.

Hongzhi Zhang, Jia Fu, Jingyuan Zhang, Kai Fu, Qi Wang, Fuzheng Zhang, and Guorui Zhou. Rlep: Reinforcement learning with experience replay for llm reasoning, 2025. URL https://arxiv.org/abs/2507.07451.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, et al. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026.

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

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025a.

Xuhui Zheng, Kang An, Ziliang Wang, Yuhang Wang, and Yichao Wu. StepSearch: Igniting LLMs search ability via step-wise proximal policy optimization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21816–21841, 2025b.

A Format-Preserving Prompt Augmentation

A critical implementation detail concerns how retrieved skills are incorporated into the prompt. Naively prepending skill text directly into the observation content disrupts the format the model expects, causing progressive degradation of valid action generation as the bank grows. The policy π_θ is well-calibrated for observations in the expected clean format (i.e., the observation distribution $\mathcal{O}_{\text{clean}}$ seen during pretraining and supervised fine-tuning); when the observation is corrupted with arbitrary preamble text, the distribution shift increases with the length of the injected text, producing increasingly malformed actions.

SAPO addresses this through format-preserving augmentation. When skills ε_t are available, the prompt is constructed as

$$\text{augment}(o_t, \varepsilon_t) = [\text{sys}(\varepsilon_t),\ \text{obs}(o_t)], \tag{9}$$

where $\text{sys}(\varepsilon_t)$ places the retrieved skills in the system role and $\text{obs}(o_t)$ preserves the observation in the user role exactly as the model expects. This separation ensures that skill injection acts as a small perturbation to the policy rather than a format-breaking corruption. The skill text is constrained to a token budget $B_{\max}$ to avoid consuming excessive prompt context.

B Analysis: Step-Level vs. Task-Level Retrieval

The skill bank reduces uncertainty about optimal actions at each decision point. Letting $H(a_t^* \mid o_t)$ denote the entropy of the optimal action distribution at step t given observation o_t, access to relevant skills ε provides information gain

$$I(a_t^*; \varepsilon \mid o_t) = H(a_t^* \mid o_t) - H(a_t^* \mid o_t, \varepsilon) > 0. \tag{10}$$

The key distinction lies in what skills are retrieved. Task-level retrieval conditions on the initial task description d, retrieving $\varepsilon_{\text{task}} = \text{retrieve}(\mathcal{B}, d)$ once for the entire episode. Step-level retrieval conditions on the current observation, retrieving $\varepsilon_t = \text{retrieve}(\mathcal{B}, o_t)$ at each step. Since o_t provides more specific context about the agent's current situation than the initial task description, step-level retrieval is expected to yield more relevant skills:

$$I(a_t^*; \varepsilon_t \mid o_t) \geq I(a_t^*; \varepsilon_{\text{task}} \mid o_t). \tag{11}$$

This advantage becomes increasingly pronounced as the episode progresses and o_t diverges from the initial context. By step 20 of a household navigation task, the agent's current room and available objects bear little resemblance to the original goal specification, rendering task-level skills increasingly irrelevant. We validate this hypothesis empirically through ablation studies comparing step-level, task-level, and hybrid retrieval modes (Section 3.2, Figure 3a).

C Benchmarks

ALFWorld [Shridhar et al., 2021] is an embodied decision-making benchmark where agents must complete multi-step household tasks through textual interactions with a simulated environment. Tasks require long-horizon planning, state tracking, and grounding actions in partially observable settings.

WebShop [Yao et al., 2022] is a web-based interactive environment that evaluates agents in realistic online shopping scenarios. The agent navigates a simulated HTML interface to search for, filter, and purchase products based on natural language instructions. The environment contains over 1.1M products and 12K instructions, yielding a large and diverse action space.

We further evaluate on *search-augmented QA tasks*, which require agents to iteratively retrieve and reason over external information. These include *single-hop* datasets (NQ [Kwiatkowski et al., 2019], TriviaQA [Joshi et al., 2017], PopQA [Mallen et al., 2023]), where answers can typically be found within a single document, and *multi-hop* datasets (HotpotQA [Yang et al., 2018], 2Wiki [Ho et al., 2020], MuSiQue [Trivedi et al., 2022], Bamboogle [Press et al., 2023]), which require aggregating evidence across multiple sources. These benchmarks test both retrieval quality and multi-step reasoning.

D Hyperparameter Configuration

Table 6 lists the full hyperparameter configuration for SAPO.

E Training Curves

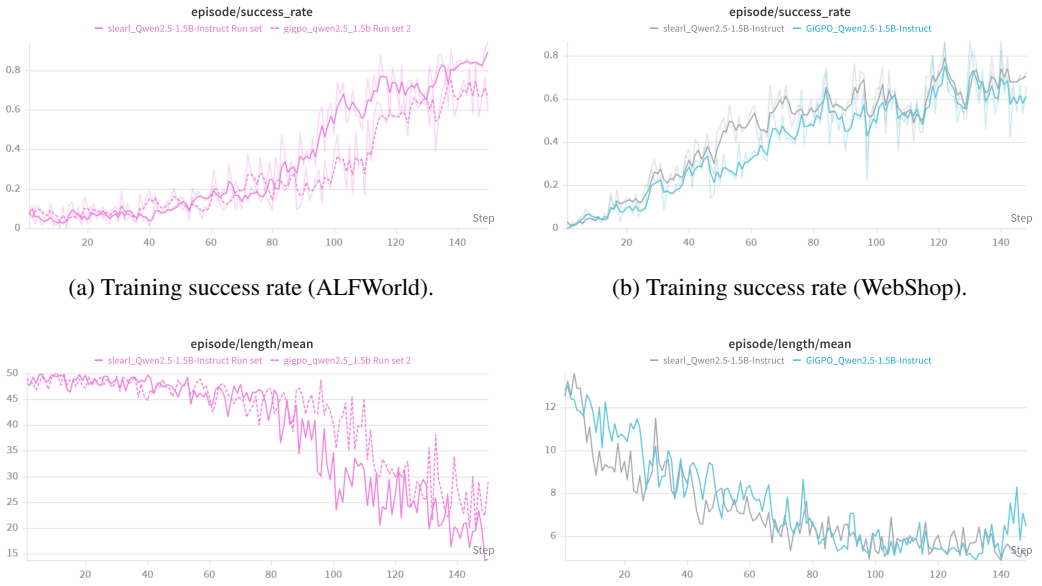
Figures 1a and 1b show the validation success rates on ALFWorld and WebShop with Qwen2.5-1.5B-Instruct. SAPO achieves both faster convergence and higher final performance compared to GRPO. On ALFWorld, SAPO begins to separate from GRPO around step 50 and reaches over 85% success by step 140, while GRPO plateaus near 63%. On WebShop, the separation is similarly pronounced: SAPO converges to approximately 74% success, compared to 53% for GRPO.

Training episode success rate. Figures 5a and 5b show the training episode success rate on ALFWorld and WebShop. On ALFWorld, SAPO exhibits a steeper rise starting around step 60, reaching over 80% by step 130, while GiGPO trails by 10–15%. On WebShop, SAPO pulls ahead around step 40 and reaches 70–75% success by step 140, compared to GiGPO's 60%. The widening gap in the mid-to-late training regime confirms that skill retrieval provides increasingly useful guidance as the bank accumulates higher-quality entries.

Episode length dynamics. Figures 5c and 5d show the mean episode length. A decreasing length indicates more efficient task completion. On ALFWorld, both methods begin near 48 steps (the

Table 6: SAPO hyperparameter configuration.

Hyperparameter	**Value**
Step-Level Retrieval	
Retrieval mode	clustered
Top-k skills	3
Max skill tokens	200
Similarity threshold δ	0.85
Warmup and Quality Gates	
Warmup epochs W	5
Min bank size $C_{\min}$	10
Self-Evolving Skill Bank	
Skill bank capacity C	100
Max skills per step K_{skill}	15
Top-k trajectories K_{traj}	5
Novelty threshold	0.85
Policy Optimization (GiGPO)	
Step advantage weight w	1.0
Advantage mode	mean_norm



(a) Training success rate (ALFWorld).

(b) Training success rate (WebShop).

(c) Mean episode length (ALFWorld).

(d) Mean episode length (WebShop).

Figure 5: Training dynamics on ALFWorld (left) and WebShop (right) with Qwen2.5-1.5B-Instruct. **Top row**: training episode success rate. SAPO achieves higher success than GiGPO throughout training, with the gap widening as the skill bank matures. **Bottom row**: mean episode length. SAPO reduces episode length faster, indicating more efficient task completion via skill-guided action selection.

maximum); SAPO drops below 20 by step 140 while GiGPO remains around 25–30, suggesting that retrieved strategies help the agent avoid unnecessary exploration. On WebShop, both converge to 5–6 steps, but SAPO reaches this efficiency earlier, particularly in the mid-training regime (steps 30–80).

SAPO System Prompts

Prompt A: Step-Level Skill Augmentation (prepended to observation at each step)

```
Below are relevant skills retrieved from the skill bank. Use them to inform your
decisions, but always prioritize the current observation.

[Relevant Skills]
{retrieved_skills}

---

Now proceed with the task. Your current observation follows:
{current_observation}
```

Prompt B: Contrastive Skill Synthesis (applied jointly to successful and failed trajectories)

```
You are analyzing both successful and failed agent trajectories sampled from the same
task batch.

Successful trajectories (reward > batch median):
{success_trajectories}

Failed trajectories (reward ≤ batch median):
{failed_trajectories}

Instructions:
1. Compare the two sets and identify the decision points that distinguished success
from failure.
2. For each key distinguishing decision, formulate a forward-looking, actionable
skill at the appropriate abstraction level:
- Principle: high-level rule that generalizes across this kind of situation (e.g.,
"Check the most likely location first")
- Pattern: reasoning chain or action template (e.g., "For heating tasks: find item
→ pick up → heat → place")
- Example: concrete instruction tied to a specific observed state (e.g., "Eggs are
commonly found in the fridge")
3. Express each skill as a positive instruction ("in state X, do Y"); never phrase
as a prohibition.
4. Output at most K_skill skills, prioritizing the highest-confidence distinguishing
decisions.
```

Figure 6: System prompts used in the SAPO pipeline. **Prompt A** augments each step's observation with retrieved skills. **Prompt B** contrastively synthesizes forward-looking skills from paired successful and failed trajectories in a single call, emitting unified entries into the skill bank $\mathcal{B}$. These prompts enable semantic skill-bank evolution without gradient updates.

F System Prompts

Figure 6 presents the two system prompts used in the SAPO pipeline. **Prompt A** is prepended to the agent's observation at each step when skill retrieval is active, providing retrieved skills via format-preserving augmentation (Section A). **Prompt B** is used during the skill-evolution phase: it receives the top-K_{traj} successful trajectories and the bottom-K_{traj} failed trajectories from the same batch in a single call, and synthesizes forward-looking skills by contrasting the two—identifying decisions that distinguished successes from failures and emitting unified, positively phrased entries that flow into the skill bank in the same format regardless of source. Together, these prompts enable semantic skill-bank evolution without gradient updates.

G Extended Related Work

This section provides a more comprehensive discussion of the relationship between SAPO and existing methods.

Comparison with SkillRL. SkillRL [Xia et al., 2026] is the most closely related skill-augmented RL method. It maintains a skill bank of reusable strategies and retrieves relevant skills to augment

the agent's prompt during RL training. However, SkillRL operates at the *task level*: skills are retrieved once based on the initial task description and held constant throughout the episode. This design is adequate for tasks where the context remains stable, but becomes a limitation in multi-turn environments where the agent's observation evolves at each step. For instance, in ALFWorld, the agent may begin in a kitchen but navigate to a bedroom mid-episode; task-level skills about "kitchen tasks" become irrelevant. SAPO addresses this through step-level retrieval conditioned on the current observation o_t, ensuring that retrieved skills remain relevant as the episode progresses. Additionally, SkillRL does not incorporate step-level credit assignment, relying solely on episode-level advantages. SAPO combines step-level retrieval with GiGPO-style step-level advantage estimation, providing fine-grained credit that identifies which specific actions benefit from which skills.

Comparison with MemRL. MemRL [Zhang et al., 2026] integrates memory mechanisms directly into the RL optimization loop. Unlike SAPO, which maintains an external skill bank that evolves through semantic analysis, MemRL updates its memory bank through gradient-based optimization. While this approach allows end-to-end training, it conflates memory quality with policy quality—if the policy degrades, memory updates may also degrade. SAPO decouples these concerns: the skill bank evolves through quality-controlled semantic extraction (score-based admission, novelty checking), independent of gradient updates, providing more stable knowledge accumulation.

Comparison with prompt-based memory methods. Methods such as Mem0 [Chhikara et al., 2025], ExpeL [Zhao et al., 2024], MemP [Fang et al., 2025], and SimpleMem [Liu et al., 2026] maintain external memory stores that guide agent behavior through in-context retrieval. These approaches operate without parameter updates, relying entirely on the base model's ability to leverage retrieved context. While effective for rapid adaptation, they cannot fundamentally improve the policy's decision-making capabilities. Hybrid approaches (Mem0+GRPO, SimpleMem+GRPO) combine memory with RL training but still retrieve at the task level. SAPO differs by (i) performing step-level retrieval conditioned on the evolving observation, (ii) using quality-controlled skill-bank evolution with score-based admission rather than unconstrained memory accumulation, and (iii) integrating multi-level credit assignment that exploits the same observation clustering used for retrieval.

Comparison with search-augmented RL. For search-augmented QA [Kwiatkowski et al., 2019, Joshi et al., 2017, Mallen et al., 2023, Yang et al., 2018, Ho et al., 2020, Trivedi et al., 2022, Press et al., 2023], Search-R1 [Jin et al., 2025] trains agents to iteratively search and reason using RL with verifiable rewards. ZeroSearch [Sun et al., 2025] and StepSearch [Zheng et al., 2025b] extend this paradigm with zero-shot and step-level search strategies, respectively. EvolveR [Wu et al., 2025] introduces evolving retrieval during training. These methods focus on improving the retrieval component of the agent, whereas SAPO augments the agent with accumulated *skills*—distilled from both successful and failed episodes—providing complementary guidance that helps the agent reason more effectively over retrieved documents.

H Limitations

SAPO has several limitations that scope its empirical claims. The framework adds compute overhead from per-step retrieval and a per-batch skill-extraction LLM call, which gives a $2.1\times$ wall-clock cost on ALFWorld and a $0.7\times$ cost on WebShop relative to GiGPO (Appendix K.4); the cost ratio is therefore environment-dependent. Main-table numbers (Tables 1, 2) are reported from a single training run per setting; multi-seed results are available only for the principal comparison with GiGPO on ALFWorld and WebShop (Table 3), where four of the six margins fall within run-to-run variation, so the smaller margins should be read with seed-level uncertainty in mind. The skill bank inherits the quality of the base-model extractor and of the underlying trajectories, and SAPO's gains are not fully internalized into policy weights—removing retrieval at evaluation drops performance below the no-skill GRPO baseline (Appendix K.1). Finally, on small or atypical benchmarks (e.g., Bamboogle), step-level skill retrieval is less effective than alternative grouping strategies (Table 2). The δ study (Table 4) covers three thresholds on one benchmark and backbone and shows that performance is sensitive to this choice; a finer sweep across benchmarks and a sweep over the top-k retrieval depth are left to future work.

Table 7: Performance on ALFWorld and WebShop. We report average success rate (%) for ALFWorld subtasks and overall, and both average score and success rate (%) for WebShop. *: results from Feng et al. [2025]. Best and second best in **bold** and underline.

		ALFWorld							WebShop	
Type	**Method**	Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Succ.
Closed-Source LLMs										
	GPT-4o	75.3	60.8	31.2	56.7	21.6	49.8	48.0	31.8	23.7
	Gemini-2.5-Pro	92.8	63.3	62.1	69.0	26.6	58.7	60.3	42.5	35.9
Qwen2.5-7B-Instruct: Prompt / Memory Methods										
	ReAct*	48.5	35.4	34.3	13.2	18.2	17.6	31.2	46.2	19.5
	Reflexion*	62.0	41.6	44.9	30.9	36.3	23.8	42.7	58.1	28.8
	Mem0	54.0	55.0	26.9	36.4	20.8	7.69	33.6	23.9	2.00
	ExpeL	21.0	67.0	55.0	52.0	71.0	6.00	46.3	30.9	11.2
	MemP	54.3	38.5	48.1	56.2	32.0	16.7	41.4	25.3	6.40
	SimpleMem	64.5	33.3	20.0	12.5	33.3	3.84	29.7	33.2	8.59
Qwen2.5-7B-Instruct: RL Methods										
	RLOO*	87.6	78.2	87.3	81.3	71.9	48.9	75.5	80.3	65.7
	GRPO*	90.8	66.1	89.3	74.7	72.5	64.7	77.6	79.3	66.1
	GiGPO*	97.7	82.7	98.8	83.7	89.3	79.2	90.8	84.4	72.8
	MemRL	62.8	38.5	22.2	12.5	8.00	0.00	21.4	29.5	9.20
	Mem0+GRPO	78.1	54.8	56.1	31.0	65.0	26.9	54.7	58.1	37.5
	SimpleMem+GRPO	89.5	36.3	60.0	50.0	64.9	26.3	62.5	67.8	46.9
	SkillRL	97.9	71.4	90.0	90.0	95.5	87.5	89.9	85.2	72.7
	SAPO	**98.5**	**85.1**	**99.0**	**93.8**	**96.4**	**90.4**	**93.5**	**87.6**	**76.3**
Qwen2.5-1.5B-Instruct										
	GRPO*	85.3	53.7	84.5	78.2	59.7	53.5	72.8	75.8	56.8
	GiGPO*	94.4	67.5	94.8	94.4	79.8	76.4	86.7	83.1	65.0
	SAPO	92.6	**80.0**	96.2	94.7	**95.0**	76.5	**87.5**	**88.7**	**75.4**

I Broader Impact and Potential Risks

SAPO improves the ability of agents to learn from accumulated skills, which can benefit applications requiring multi-step reasoning and decision-making, such as virtual assistants, scientific discovery, and automation. However, the framework may also amplify biases or errors present in the collected trajectories, as these are distilled into reusable skills and reused across tasks. In addition, skill reuse could lead to over-reliance on past patterns, reducing adaptability in novel or adversarial settings. The increased computational cost of step-level retrieval may further limit accessibility. Careful curation of skill-extraction data, monitoring of failure cases, and efficiency improvements are important to mitigate these risks.

J Full Main Results Tables

K Additional Ablations and Sensitivity Analysis

This appendix collects additional ablations and sensitivity analyses that further characterize SAPO's components and hyperparameters. All experiments use Qwen2.5-7B-Instruct unless noted otherwise. As in the main tables, each configuration is run once per setting because of the high per-run compute cost of multi-turn agent RL (Appendix K.4); multi-seed results for the principal comparison with GiGPO are reported in Table 3.

K.1 Runtime Retrieval Dependence: Removing Retrieval at Inference

Table 9 measures the dependence of the trained SAPO policy on runtime retrieval. We compare the full method against a variant trained identically but evaluated with retrieval disabled (the prompt augmentation $\mathrm{augment}(o_t, \varepsilon_t)$ is replaced with the raw observation o_t). Removing skills at evaluation drops performance to 53.6 / 61.2, well below both the full SAPO (93.5 / 76.3) and even the skill-free GRPO baseline (77.6 / 66.1). This indicates that SAPO's training is tightly coupled to the skill-augmented context: the policy learns to attend to and act on retrieved skills as an integrated part of its decision-making, rather than internalizing skill knowledge as standalone behavior. Pulling the

Table 8: Performance on search-augmented QA tasks. SAPO is trained on NQ and HotpotQA. †: in-domain datasets. Best and second best among RL methods in **bold** and underline.

		Single-Hop QA			Multi-Hop QA				
Type	**Method**	NQ†	TriviaQA	PopQA	HotpotQA†	2Wiki	MuSiQue	Bamboogle	Avg.
Qwen2.5-7B-Instruct: Prompting Methods									
	Qwen2.5	11.6	35.6	14.4	1.20	16.4	4.80	22.2	15.2
	CoT	12.8	35.6	24.0	3.80	16.2	6.60	22.6	17.4
	RAG	27.4	58.2	16.8	17.8	25.8	9.40	23.2	25.5
	Search-o1	19.4	40.6	30.4	11.4	17.0	8.60	27.0	22.1
	R1-Instruct	21.0	44.9	19.2	17.1	20.8	6.00	27.5	22.4
	Search-R1	39.3	61.0	36.8	39.7	37.0	14.6	40.1	38.5
	ZeroSearch	43.6	61.8	27.8	51.5	34.6	18.4	35.2	39.1
	EvolveR	43.5	63.4	54.4	44.6	38.2	15.6	42.0	43.1
Qwen2.5-7B-Instruct: RL Methods									
	RLOO	40.7	72.5	43.1	49.6	55.0	62.2	24.8	49.7
	PPO	38.7	75.4	48.7	48.6	59.7	63.4	26.2	51.5
	GRPO	40.3	77.0	49.6	48.9	57.7	65.1	25.0	51.9
	Reinforce++	34.3	67.5	44.3	45.9	54.5	61.2	23.7	47.3
	GSPO	41.5	77.7	45.4	46.3	60.1	67.6	25.4	52.0
	GiGPO	46.4	64.7	46.1	41.6	43.6	18.9	68.9	47.2
	IGPO	46.7	80.1	52.5	57.2	68.2	31.4	**74.9**	58.7
	SkillRL	45.9	63.3	**73.8**	45.9	43.2	20.2	40.3	47.1
	SAPO	**48.5**	**81.8**	55.2	**59.8**	**70.5**	**77.2**	33.2	**60.9**

skill prompts out at evaluation introduces a train/test distribution shift, pushing the policy below the no-skill baseline. Runtime retrieval is therefore not auxiliary scaffolding that can be discarded after training; it is an integral component of the trained policy.

Table 9: Removing retrieval at evaluation (Qwen2.5-7B-Instruct).

Variant	**ALFWorld All**	**WebShop Succ.**
GRPO (no retrieval, no clustering)	77.6	66.1
SAPO − eval retrieval (trained w/ retrieval, evaluated w/o)	53.6	61.2
SAPO (train + eval w/ retrieval)	93.5	76.3

K.2 Cluster-Indexed vs Full-Library Retrieval

Table 10 and Figure 7 compare cluster-indexed retrieval against full-library top-k retrieval (which scans all $|\mathcal{B}|$ entries) and a random-retrieval sanity check.

Table 10: Retrieval-mechanism ablation (Qwen2.5-7B-Instruct).

Variant	**ALFWorld All**	**WebShop Score**	**WebShop Succ.**
Random retrieval (sanity)	76.4	78.8	65.7
Full-library top-k (no clustering)	89.4	84.9	72.2
Cluster-indexed (ours)	93.5	87.6	76.3

K.3 Skill Bank Dynamics over Training

Figure 8 tracks two statistics of the self-evolving skill bank from Qwen2.5-1.5B-Instruct runs on ALFWorld and WebShop, logged every training step. The left panel reports the per-batch *trajectory success ratio*: as the policy improves under SAPO's joint training, success rates rise on both benchmarks (from below 5% to above 90% on ALFWorld and similarly on WebShop), providing an increasingly informative pool of trajectories for skill extraction. The right panel reports the number of skills admitted to the unified bank $\mathcal{B}$ per training step. On both benchmarks, after an initial burn-in where the score-based admission rule freely accepts new candidates (the bank is far from capacity), the admission rate decays as the bank fills up and the lowest-scoring entries become harder to displace, consistent with the bank converging on a stable set of high-utility skills rather than churning indefinitely.

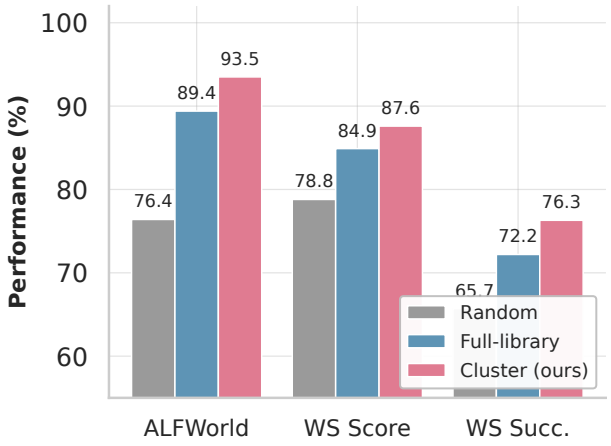


Figure 7: Retrieval-mechanism ablation on Qwen2.5-7B-Instruct (visualization of Table 10). Cluster-indexed retrieval (ours) outperforms both full-library top-k scanning and a random-skill sanity check on all three metrics.

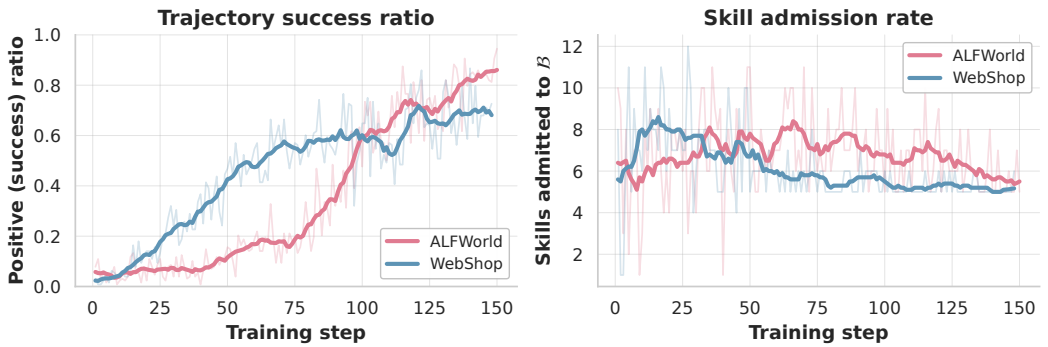


Figure 8: Skill bank dynamics on ALFWorld and WebShop with Qwen2.5-1.5B-Instruct. **Left**: per-batch trajectory success ratio; the rising curves indicate the policy is producing increasingly successful rollouts as training progresses on both benchmarks. **Right**: number of skills admitted to the unified bank $\mathcal{B}$ per training step; the admission rate decays as the bank fills up, indicating convergence rather than unbounded churn. Faint traces show raw values; bold lines are 10-step rolling means.

K.4 Training Cost

Table 11 reports the wall-clock training time of SAPO and GiGPO on Qwen2.5-1.5B-Instruct, with multipliers measured relative to GiGPO. SAPO incurs additional cost on ALFWorld due to per-step skill retrieval and the auxiliary skill-extraction call, but on WebShop it is slightly faster than GiGPO; the path-shortening effect of skill-guided action selection reduces total per-step rollout cost (see Figure 5), offsetting the retrieval overhead.

Table 11: Training cost comparison on Qwen2.5-1.5B-Instruct (wall-clock hours). Multipliers are relative to GiGPO on the same benchmark.

Method	ALFWorld	WebShop
GiGPO	18.0 h (1.0×)	11.0 h (1.0×)
SAPO	38.0 h (2.1×)	8.0 h (0.7×)

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Introduction describe SAPO's three contributions—step-level skill retrieval with observation clustering, a self-evolving bipolar skill bank, and observation-cluster-indexed step-level credit assignment—which exactly match the framework in Section 3 and the empirical results in Section 4 (93.5% on ALFWorld, 76.3% success on WebShop, 60.9% average across seven QA tasks).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix H is a dedicated Limitations section that discusses the per-step retrieval and skill-extraction compute overhead (with wall-clock numbers from Appendix K.4), the single-run main-table numbers (with multi-seed results only for the principal GiGPO comparison) and the resulting seed-level uncertainty, the dependence of the skill bank on the base-model extractor's quality, the upper bound on bank quality imposed by the underlying trajectory quality, the underperformance on small/atypical benchmarks (Bamboogle), the coupling between training and inference retrieval, and the limited scope of the δ sensitivity study (three thresholds on WebShop) together with the missing top-k sweep.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not contain formal theorems, lemmas, or proofs. The information-gain comparison in Appendix B is presented as an informal motivating argument, not as a formal theoretical result.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 3 specifies the SAPO algorithm (skill bank structure, cluster-indexed retrieval, self-evolving update rules, multi-level objective in Eq. 8); Section 4.1 reports the base models, optimizer, learning rate, batch size, group size, clipping/KL coefficients, similarity threshold, warmup, and retrieval top-k; Appendix D lists the full hyperparameter table; Appendix F provides the exact skill-extraction system prompts; and Appendix C documents all benchmarks used.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The SAPO training code, configuration files, and launch scripts (as an extension of the open-source verl-agent framework) are publicly available at `https://github.com/kingofspace0wzz/slea-rl` (also linked from the abstract), together with environment-setup instructions for ALFWorld, WebShop, and the search-augmented QA pipeline. All benchmark datasets and base models used (Qwen2.5-1.5B/7B-Instruct) are publicly available; the corresponding sources are cited in Section 4 and Appendix C.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4.1 reports the optimizer settings (learning rate 1×10^{-6}, batch size 16, group size G=8, gradient accumulation 4, clipping ϵ=0.2, KL coefficient β=0.01, step-advantage weight w=1.0, similarity threshold δ=0.85, warmup W=5 epochs, and top-3 skills retrieved per step from the unified skill bank), and Appendix D (Table 6) provides the full configuration including skill-bank capacity C, max-skill-token budget, max skills per step K_{skill}, novelty threshold, and top-k trajectories used for extraction.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Table 3 reports mean $\pm$ standard deviation over three random seeds for the principal comparison between SAPO and GiGPO on ALFWorld and WebShop at both model scales (GiGPO statistics from Feng et al. [2025]), and Section 4.2 states which differences exceed run-to-run variation. Because multi-turn agent RL training is computationally expensive (each run involves thousands of long-horizon episodes across multiple environments), the remaining numbers in Tables 1 and 2 and the ablations are reported from a single training run per setting, as stated in Section 4.2 and Appendix H. Training-curve figures in Appendix E additionally show raw traces alongside smoothed lines to give a visual sense of step-level variance during training.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Appendix K.4 reports per-run wall-clock training time on Qwen2.5-1.5B-Instruct (GiGPO 18.0 h vs. SAPO 38.0 h on ALFWorld; 11.0 h vs. 8.0 h on WebShop), and Appendix K.2 specifies the GPU type (NVIDIA A100) used for the retrieval-latency comparison. The current draft does not yet report the GPU count per run, GPU memory, or a project-wide compute estimate including preliminary and ablation runs; we will add these in the revised version.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses publicly available academic benchmarks (ALFWorld, WebShop, NQ, TriviaQA, PopQA, HotpotQA, 2Wiki, MuSiQue, Bamboogle), trains on open-weight models (Qwen2.5-1.5B/7B-Instruct), involves no human subjects or personally identifiable data, and conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix I discusses positive impacts (more capable assistants for multi-step reasoning, scientific discovery, and automation) and potential negative impacts (amplification of biases present in collected trajectories when distilled into reusable skills, over-reliance on past patterns reducing adaptability in novel or adversarial settings, and reduced accessibility from increased compute cost), along with mitigation directions such as careful curation of skill-extraction data and monitoring of failure cases.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: SAPO is a training framework rather than a new high-risk model or scraped dataset. It fine-tunes existing open-weight LLMs (Qwen2.5) on standard public agent and QA benchmarks; any released checkpoints inherit the safety profile of the base model and the constrained agent action spaces of ALFWorld/WebShop/search environments, and we do not release any new scraped data.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets (ALFWorld, WebShop, NQ, TriviaQA, PopQA, HotpotQA, 2Wiki, MuSiQue, Bamboogle), base models (Qwen2.5-1.5B/7B-Instruct), and infrastructure code (the verl-agent training framework, including the GiGPO baseline whose results we cite) are publicly released academic resources; the original papers are cited in Section 4 and Appendix C, and we use each asset under its respective open license (Apache-2.0 / MIT / dataset-specific research licenses) without redistributing the underlying data.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

Justification: The new assets are the SAPO training code, configuration files, and skill-extraction system prompts (Appendix F), released at `https://github.com/kingofspace0wzz/slea-rl`. The release includes documentation describing the algorithm, supported environments, hyperparameter configurations (Appendix D), and reproduction instructions; the codebase states its license alongside the upstream verl-agent license.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects. All experiments are conducted in simulated agent environments (ALFWorld, WebShop) or on offline QA datasets, and skill extraction is performed entirely by the same base LLM used for rollout.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve human subjects, so IRB approval is not required.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: An LLM is a core methodological component of SAPO: the same base policy model is used to contrastively analyze paired successful and failed trajectories and synthesize forward-looking entries for the unified skill bank $\mathcal{B}$ during self-evolving skill-bank updates (Section 3.4). The exact extraction prompts are provided in Appendix F. This LLM-driven semantic analysis—rather than gradient updates—is what evolves the skill bank, and is distinct from any writing/editing assistance.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.