PLUME: PERSISTENT LEARNING FROM UNIFIED MULTI-ROLLOUT EXPERIENCE FOR AGENTIC REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models are increasingly trained to act in multi-step agent environments, and group-relative policy optimization (GRPO) with verifiable rewards is widely used to train such agents. However, its trajectory-level advantages provide limited guidance on which decisions contributed to the outcomes, and the sampled trajectories are discarded after policy updates without their textual experience being distilled into reusable lessons. We introduce PLUME (Persistent Learning from Unified Multi-Rollout Experience), a framework that learns from rollout groups not only in numerical space through GRPO's relative advantages but also in text space through contrastive analysis. Within each task, PLUME compares alternative trajectories to extract lessons from successes and failures. It then compares these task-specific lessons across a batch to distill generalizable skills, which are accumulated in a persistent skill bank across training steps. Both trajectory analysis and skill generalization are performed by the policy model itself, requiring neither a stronger external teacher nor externally supplied skills. The accumulated skills guide policy optimization through self-distillation, without being inserted into the policy's rollout or inference-time prompts. PLUME thus yields two artifacts: an improved policy and a standalone skill bank for transfer. Experiments on ALFWorld, WebShop, and search-augmented QA show consistent gains across three model scales. The resulting skill bank also transfers zero-shot to larger models from different families, improving performance in almost all evaluations across the three benchmarks.

1 INTRODUCTION

Large language models are increasingly used as agents in multi-step environments, where they interact with the environment through sequences of actions and observations (Jin et al., 2025; Zhang et al., 2025a). Reinforcement learning with verifiable rewards is widely used to train such agents (Jin et al., 2025; Feng et al., 2025). A common approach is group-relative policy optimization (GRPO), which samples multiple trajectories for each task and computes relative advantages from their rewards to update the policy (Shao et al., 2024).

These trajectories contain more than the outcomes captured by their rewards. Different attempts at the same task may reveal useful strategies, recurring mistakes, and alternative ways to recover from unsuccessful actions. Yet standard outcome-supervised GRPO uses these attempts for reward-weighted policy updates and then discards them, without explicitly extracting reusable lessons for subsequent training. Its trajectory-level advantages also provide limited guidance on individual decisions. For example, a successful trajectory may include unnecessary actions, while a failed one may contain useful intermediate steps (Yang et al., 2026b; Zheng et al., 2026). The sampled experience therefore offers an opportunity for learning beyond the scalar reward signal.

This opportunity is particularly relevant for language-model agents because their interaction trajectories can be analyzed as text. Comparing attempts at the same task can help identify what successful trajectories did differently and where failed attempts went wrong (Zheng et al., 2026). Comparing the resulting lessons across tasks can further separate task-specific details from reusable strategies, while retaining them allows later updates to draw on earlier experience (Zhang et al., 2025b; Yang

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

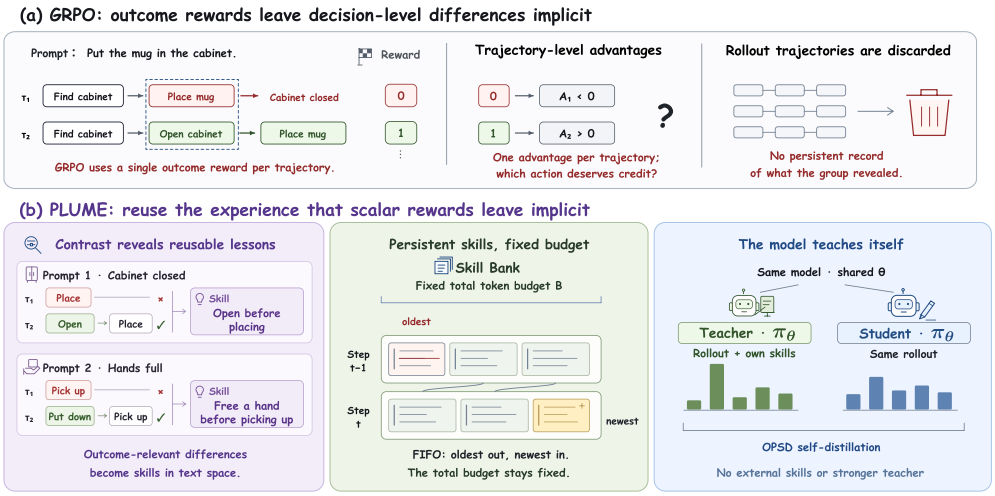

Figure 1: PLUME reuses the experience that scalar rewards leave implicit. (a) GRPO scores a trajectory with a single outcome reward, leaving which action deserved the credit implicit, and discards the trajectories without leaving a record of what the group revealed. (b) PLUME contrasts the rollouts of a group in text space into reusable skills, maintains them in a skill bank under a fixed token budget with FIFO eviction, and lets the same model teach itself: the teacher conditions on its own skills while the student re-scores the same rollout.

et al., 2026c). The challenge goes beyond producing a one-off reflection: as the policy changes, its rollout distribution and remaining errors also change. Accumulated experience must therefore be refreshed with new evidence and connected to the policy's ongoing training.

This motivates a central question: **How can an agent turn its own on-policy rollout groups into persistent guidance that evolves with the policy, without relying on externally generated skills?**

We introduce PLUME (Persistent Learning from Unified Multi-Rollout Experience), a framework that extracts and generalizes lessons from the agent's own rollout groups, retains them in a persistent skill bank, and uses the evolving bank to guide subsequent policy updates. It begins with comparisons within each task: a Group Analyzer reads alternative trajectories alongside their outcomes and extracts lessons from successful and failed attempts. Such lessons are grounded in concrete interactions, but may remain specific to the task from which they were derived. PLUME therefore compares lessons across tasks in the same batch to identify shared strategies and distill generalizable skills. These skills are then integrated into a persistent skill bank across training steps, allowing experience from earlier batches to remain available for subsequent learning. The three levels connect concrete attempts to reusable skills and, ultimately, to an accumulated record of experience.

The policy model itself performs both group analysis and skill generalization, requiring neither a stronger external teacher nor externally supplied skills. Extracting and storing skills alone does not update the policy: the lessons remain in an external text bank, while the policy must learn to apply them through its parameters. On-policy self-distillation (OPSD) (Zhao et al., 2026; Hübotter et al., 2026; Wu et al., 2026) provides a way to convert textual skills into gradient signals for updating the policy model. PLUME uses the accumulated skill bank as the privileged context for this process. The same policy serves as both teacher and student: the teacher re-scores sampled actions with the skill bank in context, while the student uses only the original interaction history. This contextual difference allows the retained lessons to provide token-level supervision that complements GRPO's trajectory-level advantages. Optimizing the two objectives jointly connects experience accumulated across training steps to updates on the current rollouts, without requiring additional environment interactions for distillation. Because the skills enter only the teacher context, the policy collects rollouts and performs inference without skill augmentation or the associated prompt overhead. Training thus produces two complementary artifacts: an improved policy that can act without consulting the

bank, and a standalone, human-readable skill bank that preserves explicit experience and can transfer to other models through prompting without additional training.

Our contributions span experience learning, agent performance, and skill transfer:

- **A self-evolving framework for joint RL and distillation.** We introduce PLUME, which turns on-policy rollout groups into a bounded persistent skill bank that guides policy updates and is refreshed by new experience from the updated policy, without externally generated skills or an external teacher.
- **Consistent gains across agent benchmarks and model scales.** Experiments across three interactive environments and three model scales demonstrate consistent gains over GRPO, while ablations support the benefits of group comparison, skill persistence, and continued skill updates.
- **Transferable experience beyond policy weights.** The learned skill bank also serves as a standalone artifact that other models can use without additional training. Prompting models from different families with the skill bank improves performance across tasks, demonstrating transfer across model sizes and families.

2 Related Work

Agentic RL and fine-grained credit assignment. Outcome-based policy optimization is widely used to train agents from environment feedback (Schulman et al., 2017; Shao et al., 2024; Jin et al., 2025; Zhang et al., 2025a). To refine coarse trajectory-level rewards, recent methods use comparisons across states or teacher–student likelihood differences to construct step-, turn-, or token-level updates (Feng et al., 2025; Yang et al., 2026a; Zheng et al., 2026). This includes deriving privileged guidance from contrasts between successful and failed rollouts (Zheng et al., 2026). The resulting signals refine credit within policy optimization. PLUME additionally retains group-derived textual experience in a skill bank across training steps, allowing accumulated skills to guide subsequent updates through auxiliary self-distillation.

Self-distillation and skill internalization. A growing line of work converts privileged textual context into dense supervision by comparing the same policy with and without feedback or skills (Zhao et al., 2026; Hübotter et al., 2026; Lu et al., 2026; Yang et al., 2026b; Wu et al., 2026). Such context can encode external knowledge or hindsight extracted from completed trajectories, providing guidance beyond scalar outcomes. A complementary line uses memory- or skill-conditioned rollouts to discover useful behavior and internalize it through cross-context policy updates (Liu et al., 2026; He et al., 2026; Tu et al., 2026). Together, these approaches establish routes from explicit experience to policy improvement without requiring the same context at deployment. PLUME focuses on generating skills through group-wise comparison and accumulating them across training steps, while collecting the actions used for distillation without skill conditioning.

Persistent experience and self-evolving agents. Language-based adaptation uses feedback and reflection to improve subsequent outputs or decisions (Shinn et al., 2023; Madaan et al., 2023; Agrawal et al., 2025). More persistent forms organize experience into evolving contextual guidance or skill libraries, enabling agents to reuse and refine knowledge across tasks (Zhang et al., 2025b; Wang et al., 2025; Xia et al., 2026; Zhou et al., 2026). Another line directly optimizes skill artifacts through execution feedback, iterative revision, and candidate evaluation (Alzubi et al., 2026; Yang et al., 2026c). Across these approaches, reusable experience is represented explicitly, with adaptation acting on context, skill artifacts, or policy parameters. PLUME couples this persistence with training-time self-distillation: its skill bank guides the teacher branch, the trained policy acts without the skill bank, and the accumulated skills remain available as a separate artifact for transfer to other models.

3 Method

PLUME recovers the information that the trajectory-level advantage discards through a second pass over the same group in text space, and turns it into a persistent, transferable artifact rather than a one-off signal. Figure 2 outlines the resulting loop. The same policy that generates $\{\tau_i\}_{i=1}^{N}$ also

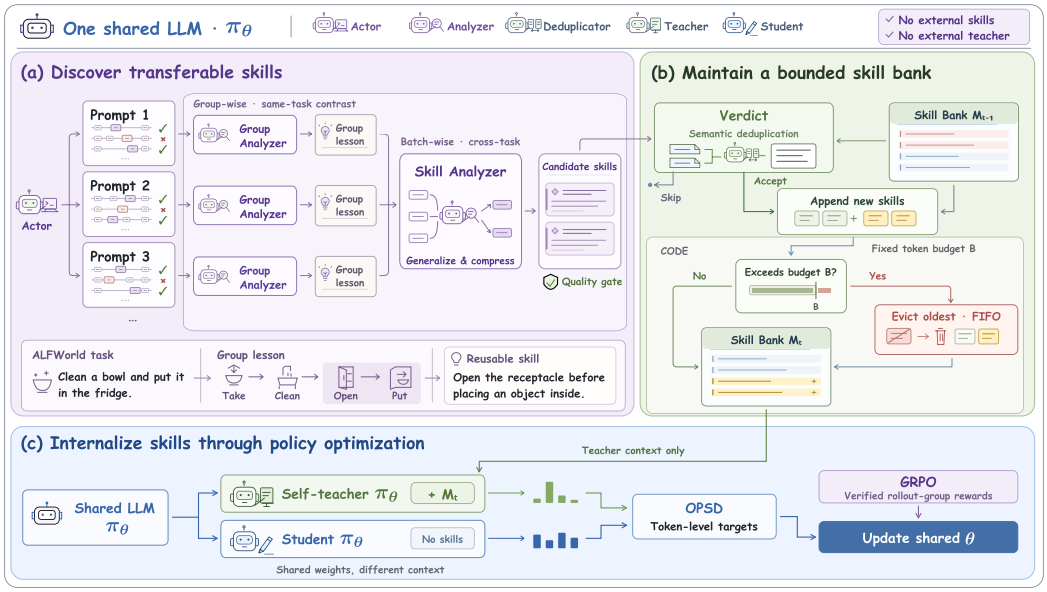

Figure 2: Overview of **PLUME**. A single LLM θ acts as the policy, the group analyzer, the skill analyzer, the deduplicator, the self-teacher, and the student. Each training step (i) contrasts the rollouts of each prompt's group into a skill, (ii) aggregates the step's skills into candidates that pass output-format checks and admits them by a verdict against the existing skill bank, (iii) updates the skill bank with deterministic code under a fixed token budget with FIFO eviction, and (iv) conditions the self-teacher on the updated skill bank while the student re-scores the same rollouts, so that OPSD's token-level targets and GRPO's group-relative loss jointly update θ.

acts as a *group analyzer*: it reads the group side by side and distills a single *skill*, subject to output-format checks—a task-specific lesson grounded in the group's actions, observations, and outcomes (Section 3.1). Skills accumulate into a *persistent skill bank* $\mathcal{B}$, maintained across the run under a capacity budget that forces older or redundant entries to retire as the policy improves (Section 3.2). The skill bank is then internalized into the policy weights through on-policy self-distillation with the skill bank as the privileged context, trained jointly with $\mathcal{L}_{\text{GRPO}}$ (Section 3.3). Because the skill bank is plain text held outside the weights, it is itself a training artifact decoupled from any specific checkpoint—a property we return to when discussing transfer to untrained models.

3.1 Group-Level Contrastive Skill Extraction

For each task x, PLUME follows GRPO (Shao et al., 2024) to sample N trajectories $\{\tau_i\}_{i=1}^N$ from $\pi_{\theta_{\text{old}}}$, with outcome rewards r_i and group-relative advantages

$$\hat{A}_i = \frac{r_i - \mu}{\sigma + \epsilon}, \qquad \mu = \frac{1}{N}\sum_{k=1}^{N} r_k, \qquad \sigma = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(r_k - \mu)^2},$$

where μ and σ are the group reward mean and standard deviation. Each advantage is broadcast to the action tokens of its trajectory. Let $h_{i,t}$ be the history before action $a_{i,t}$ and $a_{i,t,j}$ its j-th token. The clipped policy-gradient loss is

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}\left[\frac{1}{|\mathcal{A}|}\sum_{(i,t,j)\in\mathcal{A}} \min\Big(\rho_{i,t,j}\hat{A}_i,\ \text{clip}(\rho_{i,t,j}, 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}})\hat{A}_i\Big)\right], \tag{1}$$

where $\mathcal{A}$ contains valid action tokens and $\rho_{i,t,j} = \pi_\theta(a_{i,t,j} \mid h_{i,t}, a_{i,t,<j}) / \pi_{\theta_{\text{old}}}(a_{i,t,j} \mid h_{i,t}, a_{i,t,<j})$. Here, ϵ stabilizes advantage normalization and ϵ_{clip} controls ratio clipping.

Beyond the scalar advantages, the same rollout group contains evidence about which decisions led to different outcomes. PLUME uses the policy model itself as a *Group Analyzer* to compare actions

and observations across these attempts. Reading the trajectories alongside their outcomes, it returns one sentence summarizing an outcome-relevant lesson, subject to output-format checks.

What the analyzer is asked depends on the composition of the group:

- **Mixed groups**, with successes and failures of the same task in one place, are the case the method is built around. The analyzer is asked where the trajectories diverged and what the successful ones did differently there, yielding a contrastive skill: a statement of the decision that separates success from failure on this task under the current policy.
- **All-success groups** provide no contrast, but their agreement is itself informative. The analyzer is asked what the successful attempts consistently did right, yielding a positive skill that consolidates a working strategy.
- **All-failure groups** provide no success–failure contrast. The analyzer is asked to identify a shared mistake and formulate a candidate avoidance lesson.

In every case, the analyzer produces one task-specific lesson per group. These lessons form the input to the next level of analysis: comparing experience across the tasks in a training batch to distill generalizable skills.

Reading a whole group at once requires a compact rendering: raw observations restate the task, the history, and the admissible actions at every step. We therefore represent each trajectory as executed actions paired with their environment responses, writing the task description once for the group to reduce repeated context.

3.2 Cross-Task Generalization and Persistent Skill Bank

A lesson extracted by the Group Analyzer from one rollout group may depend on task-specific objects, observations, or action sequences. Such details make the lesson concrete, but can limit its relevance beyond the original task. PLUME therefore introduces a *Skill Analyzer* to perform a second level of analysis over the group-level lessons collected within the same training batch. Both analyzers use the policy model, but serve distinct roles: the Group Analyzer compares trajectories within a task, while the Skill Analyzer compares lessons across tasks to identify recurring strategies and failure patterns, separating shared principles from the details of individual attempts. The resulting skill candidates express what can carry across tasks, rather than simply concatenating the lessons of separate groups. This batch-level generalization bridges the comparison of trajectories within a task and the accumulation of reusable experience across training steps.

The persistent skill bank gives these generalized lessons a role beyond the batch that produced them. Skill admission is guided by novelty relative to the existing bank: a *Deduplicator*, also implemented by the policy model, evaluates whether each candidate contributes a lesson not already covered. Repeated experience can therefore be represented by an existing skill, while candidates that add new information can extend the bank's coverage. To keep extraction grounded in the current rollouts, the Group Analyzer extracts task-specific lessons and the Skill Analyzer generalizes them without access to the bank. Only the Deduplicator receives the bank to assess candidates for admission. This separates the evidence used to formulate a skill from the accumulated record used to judge its novelty, reducing the risk that candidate generation merely reproduces existing entries.

The bank is maintained under a fixed token budget, with skills evicted in first-in-first-out order when new entries exceed its capacity. Its contents can thus change as training encounters new experience, while the amount of text supplied to the teacher remains bounded. Persistence enables reuse across training steps rather than indefinite retention: a lesson remains available after its source trajectories leave the current batch, until the skill itself is evicted. The retained skills serve as privileged context for the self-distillation teacher, allowing updates on current rollouts to draw on lessons extracted from earlier batches. They are not inserted into the prompts used for rollout collection or inference, so the policy learns from this accumulated experience without requiring the bank during deployment.

3.3 Internalizing Experience via On-Policy Self-Distillation

Storing lessons does not itself update the policy, and GRPO's trajectory-level advantages do not directly incorporate the bank's textual guidance. PLUME uses on-policy self-distillation to turn this

guidance into token-level training signals: the same policy acts as a bank-conditioned teacher and a student using only the original interaction history. For each sampled action token, we re-score it under the student history $h_{i,t}$ and the teacher history $\tilde{h}_{i,t}$, with the bank inserted after the task description:

$$\ell^{\theta}_{i,t,j} = \log \pi_\theta(a_{i,t,j} \mid h_{i,t},\, a_{i,t,<j}), \qquad \ell^{+}_{i,t,j} = \log \pi_\theta\Big(a_{i,t,j} \mid \tilde{h}_{i,t},\, a_{i,t,<j}\Big). \tag{2}$$

We implement the OPSD auxiliary objective using the teacher–student gap gate and sampled-token loss formulation of SDAR (Lu et al., 2026). In our notation, the detached gap and gate are

$$\Delta_{i,t,j} = \mathrm{sg}[\ell^{+}_{i,t,j} - \ell^{\theta}_{i,t,j}], \qquad g_{i,t,j} = \mathrm{sigmoid}(\beta \Delta_{i,t,j}), \tag{3}$$

where sg denotes stop-gradient and β controls gate sharpness. The resulting gated OPSD loss is

$$\mathcal{L}_{\mathrm{OPSD}}(\theta) = \frac{\sum_{i,t,j} m_{i,t,j} g_{i,t,j} \left(\mathrm{sg}[\ell^{+}_{i,t,j}] - \ell^{\theta}_{i,t,j}\right)}{\sum_{i,t,j} m_{i,t,j}}, \tag{4}$$

where $m_{i,t,j} \in \{0, 1\}$ selects supervised action tokens, and the denominator normalizes by their count. On a fixed rollout batch, this yields a teacher-weighted likelihood update: gradients flow only through the student log-probability, while the teacher influences the update through the values of the detached weights $g_{i,t,j}$. Larger teacher–student log-probability gaps receive larger weights, prioritizing sampled tokens whose likelihood increases under skill conditioning. PLUME supplies its evolving skill bank as the privileged teacher context and combines this gated OPSD objective with the GRPO backbone:

$$\mathcal{L}(\theta) = \mathcal{L}_{\mathrm{GRPO}}(\theta) + \lambda\, \mathcal{L}_{\mathrm{OPSD}}(\theta), \tag{5}$$

where λ is the distillation coefficient, set to zero outside the scheduled distillation window.

The skill bank and OPSD form a write-then-rehearse loop that connects experience extraction with policy learning. The Group Analyzer extracts task-specific lessons from rollout contrasts, and the Skill Analyzer generalizes them across tasks (Sections 3.1–3.2). Retained skills condition the OPSD teacher, allowing updates on current rollouts to draw on experience from earlier batches. As new skills enter the bank, this training context evolves alongside the policy. Together, skill generation, bank maintenance, and policy optimization form a self-evolving loop that requires neither externally supplied skills nor a stronger external teacher. Because the bank is used only on the teacher side, the trained policy acts using its updated weights, with no analyzer, skill bank lookup, or augmented prompt at inference time.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate on three agentic benchmarks spanning embodied, web, and search interaction. **ALFWorld** (Shridhar et al., 2021) grounds household tasks in a text environment, requiring the agent to locate, transform, and place objects across six task types; we report success rates per task type and the micro-averaged success rate over all evaluation instances. **WebShop** (Yao et al., 2022) simulates an e-commerce site where the agent must purchase a product satisfying a natural-language instruction; we report the task score and the success rate. **SearchQA** follows the seven-dataset search-augmented QA setting of Jin et al. (2025), where agents interleave search and reasoning. It covers (i) *factual question answering*: Natural Questions (NQ) contains real search queries (Kwiatkowski et al., 2019), TriviaQA contains trivia questions (Joshi et al., 2017), and PopQA probes long-tail entity knowledge (Mallen et al., 2023); and (ii) *multi-hop reasoning*: HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023) require combining evidence across documents or composing intermediate answers. We report results on each dataset and their average.

Baselines. **Vanilla** evaluates the original Qwen3 model (Yang et al., 2025) directly, without additional training. **GRPO** (Shao et al., 2024) is the group-relative RL backbone. The remaining baselines cover the self-distillation family: **OPSD** trains an on-policy self-distillation objective alone, without the policy-gradient loss; **GRPO+OPSD** adds this objective to GRPO and trains both jointly on the same rollouts; **SDAR** (Lu et al., 2026) applies its confidence-gated form of

Table 1: Main results on ALFWorld, Search-QA, and WebShop across three model scales. Results are averaged over three seeds, with standard deviations reported in Appendix E. Within each model scale, the best result in each column is bold with light red shading, and the second-best distinct value is underlined with light blue shading; ties share the same marking.

	ALFWorld							Search-QA								WebShop	
Method	**Pick**	**Look**	**Clean**	**Heat**	**Cool**	**Pick2**	**Avg**	**NQ**	**Triv**	**Pop**	**Hotp**	**2Wk**	**MuS**	**Bam**	**Avg**	**Score**	**Succ.**
Qwen3-1.7B																	
Vanilla	21.0	30.8	3.7	2.1	4.0	5.6	10.7	22.15	33.39	21.71	17.88	18.35	3.94	14.13	22.08	44.26	4.40
GRPO	48.6	38.5	29.6	41.7	32.0	20.8	35.5	40.65	57.59	43.96	35.20	30.60	11.47	29.07	40.65	63.03	48.60
OPSD	13.3	30.8	0.0	6.2	0.0	4.2	7.6	6.42	15.62	10.46	8.01	16.34	1.24	9.07	11.95	43.2	4.2
GRPO+OPSD	54.3	46.2	42.0	29.2	40.0	20.8	40.0	40.95	**58.73**	42.91	**37.63**	33.89	12.80	30.67	41.85	72.0	48.9
RLSD	52.4	30.8	50.6	37.5	**48.0**	9.7	40.2	38.59	56.40	41.56	35.79	34.13	12.94	**35.47**	40.61	67.19	48.13
SDAR	**59.0**	53.8	46.9	**56.2**	36.0	16.7	44.5	39.31	57.31	43.66	36.53	36.68	**13.25**	30.67	42.17	68.61	52.00
PLUME (ours)	**59.0**	**56.4**	**56.8**	54.2	40.0	**48.6**	**52.62**	**41.14**	58.04	**44.66**	35.80	**37.08**	11.41	30.13	**42.64**	**74.60**	**56.40**
Qwen3-4B																	
Vanilla	59.0	30.8	18.5	22.9	6.7	22.2	28.81	30.09	49.37	31.11	25.90	28.89	7.10	28.53	32.62	11.98	1.00
GRPO	89.5	**64.1**	69.1	66.7	57.3	48.6	67.86	47.05	**64.03**	45.60	40.99	42.70	15.38	37.07	46.94	75.03	62.33
OPSD	58.1	28.2	16.0	31.2	5.3	12.5	26.90	3.32	8.73	4.44	2.60	3.90	0.46	2.93	4.72	10.6	1.5
GRPO+OPSD	**90.5**	51.3	59.3	70.8	45.3	23.6	59.0	**47.53**	63.81	**47.54**	42.75	42.28	16.94	42.40	**47.69**	80.8	69.1
RLSD	83.8	59.0	76.5	**83.3**	58.7	50.0	69.8	45.74	62.56	44.66	40.95	37.54	**18.18**	**45.07**	45.15	82.24	70.80
SDAR	77.1	56.4	72.8	66.7	57.3	50.0	65.0	45.84	61.08	43.06	40.21	40.07	17.60	42.13	44.87	78.25	66.80
PLUME (ours)	86.67	51.28	**81.48**	81.25	**73.33**	**52.78**	**73.57**	47.18	63.15	45.34	**43.15**	**44.18**	18.05	43.20	47.49	**85.51**	**72.93**
Qwen3-8B																	
Vanilla	61.9	33.3	23.5	20.8	25.3	33.3	35.71	27.03	53.01	28.26	26.73	31.36	7.61	34.93	33.17	26.53	4.33
GRPO	84.8	64.1	66.7	62.5	58.7	52.8	66.7	46.90	65.42	44.08	44.29	47.66	19.68	**48.27**	48.71	84.55	71.53
OPSD	64.8	35.9	17.3	20.8	29.3	30.6	35.7	31.03	49.25	36.22	28.43	29.28	11.16	34.40	34.73	12.6	0.9
GRPO+OPSD	83.8	53.8	74.1	66.7	66.7	68.1	71.4	48.01	**65.74**	47.42	44.29	44.86	19.67	46.67	49.10	82.1	71.4
RLSD	88.6	**66.7**	67.9	79.2	58.7	45.8	68.81	47.54	64.49	46.14	44.96	42.74	20.22	46.13	48.04	82.45	71.00
SDAR	82.9	56.4	84.0	58.3	**80.0**	61.1	73.6	47.90	64.49	45.93	43.11	42.99	19.18	45.87	47.76	81.30	71.87
PLUME (ours)	**98.1**	64.1	**95.1**	**87.5**	78.7	**73.6**	**85.48**	**49.50**	65.24	**47.70**	**45.47**	**48.06**	**21.36**	46.93	**50.20**	**86.51**	**79.67**

self-distillation; and **RLSD** (Yang et al., 2026a) uses teacher–student differences to modulate token-level RL update magnitudes. The self-distillation baselines use teacher contexts assembled from the released SkillRL library (Xia et al., 2026), whose skill-distillation pipeline uses OpenAI o3 (OpenAI, 2025). In contrast, PLUME generates and maintains its skills using the policy being optimized, without an external model for skill generation. The persistence of the skill bank is examined in Section 4.3. All methods use the same backbone models, rollout budget, and number of training steps.

Implementation. All methods are trained from Qwen3-1.7B, Qwen3-4B, and Qwen3-8B (Yang et al., 2025) in a verl-agent-based trainer with rollout group size $N=8$. The batch size is 16 tasks per step on ALFWorld and WebShop and 128 on SearchQA; on SearchQA, the skill analysis summarizes the rollouts of 16 prompts per step.

4.2 Main Results

Strong performance across model scales. Table 1 shows that PLUME achieves the highest ALFWorld average success rate and both WebShop metrics at all three model scales. At Qwen3-8B, PLUME reaches 85.48% success on ALFWorld and 79.67% on WebShop, exceeding the strongest baseline by 11.88 and 7.80 percentage points, respectively. These improvements hold under matched backbone models, rollout budgets, and training steps, and extend beyond reward-only RL to joint RL–distillation baselines.

Self-generated guidance strengthens self-distillation. PLUME adopts SDAR's confidence-gated self-distillation but constructs its guidance differently. In our experiments, SDAR and RLSD use SkillRL skills (Xia et al., 2026) distilled with OpenAI o3 (OpenAI, 2025), whereas PLUME extracts and continually updates skills using the policy itself. PLUME outperforms SDAR on ALFWorld Avg, SearchQA Avg, and both WebShop metrics at every scale, without an external skill-generation model. This suggests that guidance grounded in the learner's own experience and refreshed during training can be more useful than externally supplied skills, even when those skills are generated by a stronger model.

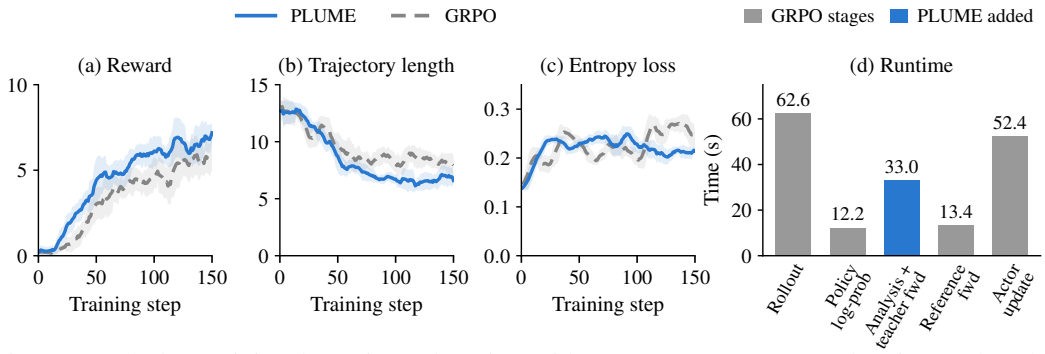

Figure 3: WebShop training dynamics and runtime with Qwen3-4B. (a–c) Reward, trajectory length, and token entropy for single PLUME and GRPO runs. (d) PLUME timings: gray denotes stages shared with GRPO; blue combines analysis, bank update, and skill-conditioned teacher computation.

Gains vary across tasks. On SearchQA, PLUME achieves the highest average at 1.7B and 8B, while ranking second at 4B, 0.20 points below GRPO+OPSD. A plausible explanation for the smaller gains is that ALFWorld and WebShop expose recurring procedures—such as ordering object manipulations and selecting product options—that can be reused across task instances. SearchQA also benefits from general search strategies, but its answers additionally require question-specific facts and retrieved evidence that reusable skills cannot replace.

4.3 Ablation Analysis

To examine the experience-based coupling proposed in Section 1, we ablate three design choices on WebShop with Qwen3-4B: comparison across trajectories, retention across training steps, and continued bank updates. Table 2 varies one component at a time.

Table 2: Ablation on WebShop with Qwen3-4B. Each row removes one component of PLUME; score and success rate are reported.

Variant	Score	Succ.
w/o group analyzer	70.78±0.48	56.33±0.81
w/o persistent skill	80.36±0.87	69.13±0.93
w/o skill update	83.04±0.17	69.60±0.33
PLUME (ours)	**85.51±0.22**	**72.93±0.38**

Group comparison. Replacing group analysis with single-trajectory analysis lowers the score from 85.51 to 70.78 and success from 72.93% to 56.33%, the largest drop among the ablations. This supports using alternative attempts at the same task to construct distillation guidance, rather than extracting each lesson from an isolated trajectory.

Skill persistence. Clearing skills after each step reduces the score by 5.15 points and success by 3.80 percentage points. Both variants use newly extracted guidance, but only the full method preserves it for later updates. The result supports the value of experience that remains available beyond its source batch, as proposed in the RL–OPSD learning loop.

Continued skill updates. Freezing the bank once it reaches its token budget reduces the score by 2.47 points and success by 3.33 percentage points. The score ordering—single-use skills (80.36), a frozen bank (83.04), and the updated bank (85.51)—supports both retaining prior experience and refreshing the guidance during training. These ablations provide evidence for the comparison, persistence, and update components in this setting; they do not directly measure the correctness of individual skills.

4.4 Interaction Efficiency

Figure 3 compares PLUME and GRPO on WebShop with Qwen3-4B under the same rollout budget. PLUME gains an early reward advantage and finishes near 7.1 versus 5.8 after 150 training steps, while producing shorter trajectories late in training. Skill extraction and distillation reuse these rollouts without additional environment interactions, but introduce auxiliary computation: analysis, bank update, and teacher computation together take 33.0 seconds, approximately 19% of the profiled

PLUME step. The interaction gains therefore do not establish a reduction in total training wall-clock time.

Table 3: Zero-shot transfer of the skill bank trained with PLUME on Qwen3-4B. Each model is evaluated with and without the skill bank in its prompt; none of these models was trained with the skill bank. ALFWorld reports success rate (%) on seen and unseen splits; WebShop reports score and success rate; Search reports the ID and OOD results. Bold marks the better value within each Base / + Skill Bank pair.

		ALFWorld		WebShop		Search	
Model	Setting	Seen	Unseen	Score	Succ.	ID	OOD
Gemini 3.5 Flash (Google, 2026)	Base	85.71	85.07	35.36	29.80	30.14	34.79
	+ Skill Bank	**92.86**	**88.81**	**39.73**	**34.40**	**30.82**	**35.62**
Qwen3-32B	Base	42.14	44.78	22.57	2.60	34.02	35.98
	+ Skill Bank	**52.14**	**58.21**	**30.69**	**17.20**	**38.81**	**39.45**
GPT-5.6-Luna (OpenAI, 2026)	Base	47.14	50.00	13.10	10.80	**36.99**	36.40
	+ Skill Bank	**73.57**	**81.34**	**22.16**	**19.80**	36.20	**36.59**

4.5 Skill-Bank Analysis and Transfer

Skill Interpretability. Beyond policy improvement, PLUME produces an inspectable record of experience. Appendix C traces task-specific lessons to retained skills: a WebShop color-selection action becomes a general option-selection rule, while a SearchQA lesson distinguishes platform release from film premiere. The retained rules omit instance-specific details, such as a product's literal option value or a film's title and release date, while preserving the decision or distinction relevant to similar tasks. This textual representation makes it possible to inspect what the bank retains and how it relates to the source lesson. We next evaluate whether this guidance can also benefit other models without parameter updates.

Skill transfer. Table 3 tests the Qwen3-4B bank in other models' prompts without updating their parameters. Gains on Qwen3-32B show that experience extracted by a smaller model can still guide a larger recipient. Improvements on ALFWorld and WebShop also extend to models from different families, supporting reuse beyond the donor's parameterization. The ALFWorld unseen split provides a further test: all three recipients improve, and two gain more on unseen than seen tasks. This is consistent with reuse of procedures across task instances, although it does not establish generalization beyond the benchmark's task structure. Transfer is also selective. Search benefits are less consistent across recipients, including a small decline for GPT-5.6-Luna. Together with the main results, this suggests that a skill's usefulness depends on how well it addresses the recipient's remaining errors: recurring interaction procedures may transfer readily, while general search advice cannot supply question-specific evidence. The bank therefore provides useful inference-time guidance independently of PLUME's weight updates; this contextual reuse is distinct from the internalization evaluated in the main results.

5 Conclusion

PLUME turns rollout experience into reusable guidance for agent training. The policy contrasts trajectories, consolidates their lessons in a persistent skill bank, and internalizes the bank through self-distillation alongside reward optimization. This produces both an improved policy that acts without skill context and an explicit record of experience that can guide other models. Across ALFWorld, WebShop, and SearchQA, the results support the value of this approach, while the ablations highlight the importance of group comparison and persistence. Gains are more consistent on interactive tasks than on search-based question answering, suggesting that the benefit of accumulated skills depends on the reusable structure of the task. Zero-shot transfer further shows that the bank's utility extends beyond its source model. Together, these findings support treating textual experience as a persistent learning resource, with future work needed to understand when extracted guidance remains useful across tasks and models.

Reproducibility Statement

We provide the experimental and implementation details needed to reproduce our results in Appendix A. This includes the training configuration, evaluation protocol, and benchmark-specific settings. Algorithm 1 summarizes the complete PLUME training procedure. Appendix B provides the prompts used for group analysis, skill generalization, and skill deduplication. Appendix E reports detailed results with means and standard deviations across three runs, while Appendix F provides additional evaluation on unseen ALFWorld environments. We also provide the implementation code in the supplementary material.

AI Use Statement

In this work, we used generative AI tools to assist with interpreting experimental results and providing feedback on experimental methodology. Additionally, we used these tools for literature search and editing, table formatting. The authors are responsible for checking AI-assisted content against the cited sources, experimental records, and implementation, and take responsibility for the final content of this work, including all text, claims, results, and artifacts produced with the aid of generative AI.

References

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning. *arXiv preprint arXiv:2507.19457*, 2025. URL https://arxiv.org/abs/2507.19457.

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. EvoSkill: Automated Skill Discovery for Multi-Agent Systems. *arXiv preprint arXiv:2603.02766*, 2026. URL https://arxiv.org/abs/2603.02766.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. In *Advances in Neural Information Processing Systems*, 2025.

Google. Gemini 3.5 Flash. Gemini API model documentation, 2026. URL https://ai.google.dev/gemini-api/docs/models/gemini-3.5-flash.

Zhongyu He, Yuanfan Li, Fei Huang, Tianyu Chen, Siyuan Chen, Xingyang Li, Meng Hsuan Yu, Xiangrong Liu, Leyi Wei, Lu Pan, Ke Zeng, and Xunliang Cai. SIRI: Self-Internalizing Reinforcement Learning with Intrinsic Skills for LLM Agent Training. *arXiv preprint arXiv:2606.02355*, 2026. URL https://arxiv.org/abs/2606.02355.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation. In *International Conference on Machine Learning (2026)*, 2026.

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 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)*, pp.

1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Zeyuan Liu, Jeonghye Kim, Xufang Luo, Dongsheng Li, and Yuqing Yang. Exploratory Memory-Augmented LLM Agent via Hybrid On- and Off-Policy Optimization. *arXiv preprint arXiv:2602.23008*, 2026. URL https://arxiv.org/abs/2602.23008.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023.

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)*, pp. 9802–9822, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.546. URL https://aclanthology.org/2023.acl-long.546/.

OpenAI. Introducing OpenAI o3 and o4-mini. Official model release, April 2025. URL https://openai.com/index/introducing-o3-and-o4-mini/.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition. Official model release, July 2026. URL https://openai.com/index/gpt-5-6/.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/.

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

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

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*, 2023.

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

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. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

Songjun Tu, Chengdong Xu, Qichao Zhang, Yiwen Ma, Yaocheng Zhang, Linjing Li, Dong Li, Xiangyuan Lan, and Dongbin Zhao. UCOB: Learning to Utilize and Evolve Agentic Skills via Credit-Aware On-Policy Bidirectional Self-Distillation. *arXiv preprint arXiv:2606.29502*, 2026. URL https://arxiv.org/abs/2606.29502.

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library. *arXiv preprint arXiv:2512.17102*, 2025.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. Seed: Self-evolving on-policy distillation for agentic reinforcement learning. *arXiv preprint arXiv:2607.14777*, 2026.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled rlvr, 2026a. URL https://arxiv.org/abs/2604.03128.

Shuo Yang, Jinyang Wu, Zhengxi Lu, Yuhao Shen, Fan Zhang, Lang Feng, Shuai Zhang, Haoran Luo, Zheng Lian, Zhengqi Wen, and Jianhua Tao. OPID: On-Policy Skill Distillation for Agentic Reinforcement Learning. *arXiv preprint arXiv:2606.26790*, 2026b. URL https://arxiv.org/abs/2606.26790.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. SkillOpt: Executive Strategy for Self-Evolving Agent Skills. *arXiv preprint arXiv:2605.23904*, 2026c. URL https://arxiv.org/abs/2605.23904.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

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

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, Yifan Zhou, Yang Chen, Chen Zhang, Yutao Fan, Zihu Wang, Songtao Huang, Francisco Piedrahita-Velez, Yue Liao, Hongru Wang, Mengyue Yang, Heng Ji, Jun Wang, Shuicheng Yan, Philip Torr, and Lei Bai. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*, 2025a.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic Context Engineering: Evolving Contexts for Self-Improving Language

Models. *arXiv preprint arXiv:2510.04618*, 2025b. URL https://arxiv.org/abs/2510.04618.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Binbin Zheng, Zijun Xie, Guanqun Zhao, Enlei Gong, Xing Ma, Xiaoliang Fu, and Zeyu Chen. Group-Reflective Self-Distillation for Agentic Reinforcement Learning. *arXiv preprint arXiv:2607.28076*, 2026. URL https://arxiv.org/abs/2607.28076.

Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, Runyu Yang, Qiangbin Liu, Xinlei Yu, Jianmin Zhou, Na Wang, Chunyang Sun, and Jun Wang. Memento-skills: Let agents design agents. *arXiv preprint arXiv:2603.18743*, 2026.

A Experimental Details

We provide the training configuration, evaluation protocol, and supplementary analyses of PLUME below.

A.1 Training Configuration

All training experiments are conducted on eight NVIDIA H800 GPUs using the verl framework. Table 4 summarizes the batch sizes, sampling settings, and token budgets used for PLUME. These settings are shared across Qwen3-1.7B, Qwen3-4B, and Qwen3-8B within each environment. Qwen3's original chat template inserts an empty `<think></think>` block when thinking mode is disabled. We observed that reusing these tags in the task prompt led to high variance and instability in training and inference results. To avoid this conflict with the chat template, we use `<reason>...</reason>` for task-level reasoning throughout training and evaluation.

Batch size counts tasks before rollout expansion.

Table 4: PLUME training configuration. Token lengths are configured upper bounds. Policy input and output limits apply to each interaction turn; analyzer limits apply to each group-analysis call. ALFWorld uses a history length of two interaction steps.

Parameter	ALFWorld	WebShop	SearchQA
Training batch size (tasks)	16	16	128
Training steps	150	150	150
Confidence-gate sharpness β	5.0	5.0	5.0
OPSD coefficient λ	0.05	0.05	0.01
Rollouts per task	8	8	8
Trajectories per training step	128	128	1,024
Rollout sampling temperature	1.0	1.0	1.0
Rollout top-p	1.0	1.0	1.0
Maximum interaction steps	50	15	4
Maximum policy input tokens	2,048	4,096	4,096
Maximum policy output tokens	512	512	512
Analyzed task groups per step	16	16	16
Analyzer sampling temperature	0.7	0.7	0.7
Maximum analyzer input tokens	16,384	16,384	16,384
Maximum analyzer output tokens	1,024	1,024	1,024
Skill-bank token budget	1,024	1,024	1,024

A.2 Evaluation Protocol

We evaluate the trained checkpoints using temperature 0.4. Table 5 summarizes the evaluation-set sizes and interaction limits.

Table 5: Evaluation settings across the three benchmarks.

Setting	ALFWorld	WebShop	SearchQA
Evaluation instances	140	512	51,713
Sampling temperature	0.4	0.4	0.4
Maximum interaction steps	50	15	4

Metrics. ALFWorld Avg is the total number of successful tasks divided by the number of evaluated tasks (micro-average). WebShop reports graded purchase score and full-task success. SearchQA reports exact match after answer normalization.

Algorithm 1 PLUME: Persistent Learning from Unified Multi-Rollout Experience

Require: Initial policy π_{θ_0}, task set $\mathcal{Q}$, group size N, bank budget K, context function H, gate sharpness β, OPSD schedule λ_k, clip parameter ϵ_{clip}, learning rate η
Ensure: Trained policy π_θ and persistent skill bank $\mathcal{B}$

$\theta \leftarrow \theta_0$; $\mathcal{B} \leftarrow \emptyset$
for training step $k = 1, \ldots, 150$ **do**
  $\theta_{\text{old}} \leftarrow \theta$; sample task batch $\mathcal{T} \subset \mathcal{Q}$
  On-policy rollout collection
  for each task $q \in \mathcal{T}$ **do**
    Sample $G_q = \{\tau_{q,n}\}_{n=1}^{N}$ with $\tau_{q,n} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$
    Compute rewards $r_{q,n}$ and group statistics μ_q, σ_q
    $\hat{A}_{q,n} \leftarrow (r_{q,n} - \mu_q)/(\sigma_q + \epsilon)$ for each trajectory
    Cache old log-probabilities ℓ^{old} of all sampled action tokens
  end for
  Group analysis and persistent bank update
  Select analysis tasks $\mathcal{T}_{\text{ana}} \subseteq \mathcal{T}$; initialize lessons $\mathcal{S} \leftarrow \emptyset$
  for each task $q \in \mathcal{T}_{\text{ana}}$ **do**
    $s_q \leftarrow \text{GroupAnalyzer}_{\theta_{\text{old}}}(q, G_q, \{r_{q,n}\}_{n=1}^{N})$
    Add (q, s_q) to $\mathcal{S}$ if the lesson passes the acceptance checks
  end for
  $\mathcal{C} \leftarrow \text{SkillAnalyzer}_{\theta_{\text{old}}}(\mathcal{S})$
  $v \leftarrow \text{Verdict}_{\theta_{\text{old}}}(\mathcal{B}, \mathcal{C})$
  $\mathcal{B} \leftarrow \text{UpdateBank}(\mathcal{B}, \mathcal{C}, v, K)$ ▷ Deduplicate; FIFO eviction
  Paired re-scoring and joint optimization
  for each optimization minibatch of sampled action tokens **do**
    Form teacher histories $\tilde{h} \leftarrow H(h, \mathcal{B})$
    Evaluate $\ell^+ \leftarrow \log \pi_\theta(a \mid \tilde{h}, a_{<j})$ and $\ell^\theta \leftarrow \log \pi_\theta(a \mid h, a_{<j})$
    $\Delta \leftarrow \text{sg}[\ell^+ - \ell^\theta]$; $g \leftarrow \text{sigmoid}(\beta\Delta)$
    $\rho \leftarrow \exp(\ell^\theta - \ell^{\text{old}})$
    Compute $\mathcal{L}_{\text{GRPO}}$ from $\rho, \hat{A}, \epsilon_{\text{clip}}$ using Eq. equation 1
    $\mathcal{L}_{\text{OPSD}} \leftarrow \mathbb{E}[m\, g\, (\text{sg}[\ell^+] - \ell^\theta)]$
    $\mathcal{L} \leftarrow \mathcal{L}_{\text{GRPO}} + \lambda_k \mathcal{L}_{\text{OPSD}}$
    $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
  end for
end for

The bank is absent from rollout collection, group and skill analysis, and student re-scoring. All sampled rollout groups contribute to the policy update; SearchQA selects 16 task groups per step for analysis. Here, m is the action-token mask and sg denotes stop-gradient. The coefficient λ_k is zero outside the OPSD training window.

B ANALYZER PROMPTS

The group analyzer receives the task, the outcomes of its rollout group, and compact action–result records. Its prompt asks for one task-specific lesson; generalization across tasks takes place in the subsequent rule-extraction stage. The prompts for ALFWorld, SearchQA, and WebShop are provided below.

B.1 GROUP ANALYZER PROMPT

B.1.1 ALFWORLD AND SEARCHQA

The instruction is selected according to the group's outcomes:

- **Mixed outcomes:** "{N} attempts at the same task: {S} succeeded and {F} failed. Write one episode_skill stating the decision that separated the successes from the failures."
- **All success:** "All {N} attempts succeeded. Write one episode_skill stating the decision that was most easily missed: the part a reasonable attempt could have gotten wrong, without which these attempts would likely have failed."
- **All failure:** "All {N} attempts failed. Write one episode_skill stating the mistake they shared."

The template below shows the shared prompt. Braced fields are filled for each group. The domain-specific field is "objects and receptacles" for ALFWorld and "entities, dates, and search terms" for SearchQA. Line wrapping is adjusted for presentation.

Group Analyzer — ALFWorld and SearchQA

```
Analyze this group of agent attempts at the same task and return ONLY
     valid JSON.

{outcome-dependent instruction}

Constraints:
- ONE sentence. State what this group teaches, not a recap of the
    steps taken.
- Write it as a standalone claim; do not reuse the wording of the
    request above.
- Naming the concrete {domain-specific items} of this task is fine --
     a later stage generalizes.
- Ground it in what the attempts show; do not claim anything the
    trajectories do not support.
- Return only episode_skill.

Return format:
{
  "episode_skill": "string"
}

Group context:
- Task: {task}
- Attempts: {N} total, {S} success / {F} failure

Trajectories (successes first; each step is action -> result):
[Trajectory 1 ({outcome}):]
  s0: {action} -> {environment result}
  ...
{remaining trajectories}
```

The task is written once, followed by trajectories ordered with successful attempts first. Each step contains the executed action and its subsequent environment observation. The response contract appears before the trajectory block so that it remains visible when long inputs require truncation.

Input example. The following excerpt illustrates the trajectory block for an ALFWorld group with two successes and six failures. Actions and observations are taken from recorded rollouts and rendered in the template's format. We show one successful and one failed trajectory; ellipses mark omitted steps and trajectories. The example illustrates the input representation, rather than isolating the cause of the outcome difference.

Input Example — ALFWorld

```
Group context:
- Task: put two watch in dresser.
- Attempts: 8 total, 2 success / 6 failure

Trajectories (successes first; each step is action -> result):
[Trajectory 1 (success):]
  s0: go to shelf 1 -> You arrive at shelf 1. On the shelf 1, you see
     a watch 2.
  s1: take watch 2 from shelf 1 -> You pick up the watch 2 from the
    shelf 1.
  ...
[Trajectory 2 (success): omitted]
[Trajectory 3 (failure):]
  ...
  s2: go to drawer 1 -> You arrive at drawer 1. The drawer 1 is
    closed.
  s3: open drawer 1 -> You open the drawer 1. The drawer 1 is open.
    In it, you see nothing.
  ...
[Trajectories 4-8 (failure): omitted]
```

The analyzer sees both action choices and their observed consequences, such as obtaining an object or finding an empty drawer. The outcome-dependent instruction and JSON response contract shown above precede this block in the full prompt.

B.1.2 WEBSHOP

For WebShop, the analyzer copies an action from the input and states when to use it. This preserves literal interface arguments such as option names. For mixed groups, the request is "Find the one action the successful attempts took that the failed ones did not, or took too late." Uniform-outcome groups instead ask for an action that made the attempts work, or one taken at the wrong time or omitted. The central instruction and output contract are:

Group Analyzer — WebShop

```
Copy that action EXACTLY as it is written below -- the action only,
    not the "-> result" part that follows it. Same words, same object
     names, same numbers. Do not paraphrase it and do not invent an
    action that does not appear.

Then say in one sentence when a future agent should take it.

Answer in exactly this form, with the two fields on separate lines
    and nothing outside the tags:

<lesson>
action: <copied exactly from the attempts below>
when: <one sentence>
</lesson>
```

B.2 SKILL ANALYZER PROMPT

The skill analyzer reads the task-specific lessons produced by the group analyzer and extracts reusable rules. Each input contains a task, its group outcome counts, and the generated lesson. The current bank is not included at this stage. The template below is shared across the three benchmarks; {domain} is replaced with "ALFWorld household tasks", "WebShop shopping tasks", or "question-answering tasks with a search tool".

Skill Analyzer — Shared Template

```
Below is evidence from several groups of agent attempts at {domain}.
Preserve the distinct reusable decisions it supports.

A rule is ONE sentence a future agent can act on. Prefer the
    condition-triggered form -- "when <situation>, do <action>" --
    over a blanket imperative; a fact about how the environment
    responds also counts as a rule. It has to be something the
    evidence below actually shows, and it has to hold for other tasks
     of the same kind, not only this one.

Account for EVERY numbered input. Combine inputs only when they teach
     the same decision: the same applicable situation, action or
    action sequence, and result.
Generalize instance names, but retain required object counts,
    prerequisites, ordering, state changes and completion conditions
    that the inputs support.
Do not replace different mechanisms with one broad reminder to act
    correctly.
A mechanism seen only once can still deserve its own rule.

Write ONE sentence per rule, as before. Do not target a fixed number
    of rules.
An input with no supported reusable decision (for example, only this
    task's answer) gets an empty skill string. Do not fill that slot
    with invented advice.

Start with <rules>, write one JSON object per line, and end with </
    rules>.
Each object has exactly two keys:
- "sources": a list of input numbers supporting this decision.
- "skill": the actual one-sentence rule supported by those inputs, or
     "" to skip.
Every input number must occur exactly once, including skipped inputs.
Numbers are bookkeeping, not part of the rule text.

The number of outputs follows the evidence: five inputs might support
     two rules and one skip; two inputs about different decisions
    might each need a rule.

ACTUAL EVIDENCE:
[1] Task: {task}  [{success count}/{group size} succeeded]
    What the attempts showed: {group-analyzer lesson}
{remaining numbered inputs}

Account for the {number of inputs} actual inputs above.
Return only the <rules> block, containing supported rules and
    explicit skips.
```

The `sources` field links each proposed skill to its supporting inputs. An empty `skill` represents an explicit skip. The instruction permits both combining evidence for the same decision and preserving a distinct rule supported by a single input.

B.3 VERDICT PROMPT

The verdict stage compares candidate skills with the existing bank. The same template is used across ALFWorld, WebShop, and SearchQA, with only the domain description and supplied rules changing. Existing bank entries are numbered; incoming candidates are lettered. The model returns coverage decisions, while code applies the corresponding bank updates.

Verdict — Shared Template

```
You keep a short list of rules for {domain}.

THE LIST AS IT STANDS:
1. {existing rule}
{remaining numbered rules}

WHAT THIS ROUND OF AGENT ATTEMPTS SUGGESTS ADDING:
A. {candidate rule}
{remaining lettered candidates}

For each suggestion, decide whether the list ALREADY says it. Two
    rules say the same thing when following either one makes you do
    the same thing; the wording does not matter. A suggestion that
    only adds detail to a rule already on the list is already said.

Answer each suggestion with the number of the rule that already
    covers it, the LETTER of an earlier suggestion that says the same
     thing, or "new" if neither.

The output form, on letters that are not yours -- answer for {
    candidate letters}, one line each:
<verdicts>
{format example generated for the number of candidates}
</verdicts>

Reply with the <verdicts> block and nothing else.
```

The verdict determines whether a candidate adds a new skill to the bank. A rule number marks it as already covered by an existing entry; an earlier candidate letter marks it as redundant within the current batch. Only candidates labeled `new` are admitted, after which deterministic code enforces the bank's token budget through FIFO eviction.

C CASE STUDIES

The examples below trace a task-specific group lesson to a more general skill retained in the bank. Task, analyzer output, and bank skill are reproduced from saved records. These examples illustrate the extraction process rather than the causal effect of a skill on subsequent performance.

WebShop: Selecting the Requested Option

TASK

Find me non slip desks for living room with color: christmasgoo3302, and size: 19.7x31.5in+19.7x63in, and price lower than 50.00 dollars

GROUP ANALYZER OUTPUT

click[christmasgoo3302] – After navigating to the product details page, the agent should click on the color option "christmasgoo3302" to ensure the correct color is selected before proceeding to buy.

BANK SKILL

when the agent is on the product details page, do click on the color option to ensure the correct color is selected before proceeding to buy.

The literal color value becomes a general option-selection rule. Here, `click[...]` denotes a structured environment action.

ALFWorld: Preserving an Action Prerequisite

TASK

find two pencil and put them in drawer.

GROUP ANALYZER OUTPUT

To successfully complete the task, the agent must first take the pencil from the desk, then move it to the drawer, and repeat the process for the second pencil.

BANK SKILL

Take the object from its current location before moving it to the target location.

The skill retains the take-before-place ordering, while omitting the task-specific requirement to repeat it for a second object.

SearchQA: Distinguishing Release Events

TASK

when does hold the dark come out on netflix?

GROUP ANALYZER OUTPUT

The most easily missed decision was recognizing that the release date on Netflix for 'Hold the Dark' is September 28, 2018, which is explicitly stated in the search results despite the film's premiere date being earlier in the same year.

BANK SKILL

when searching for the release date of a film on Netflix, recognize that the release date stated on the platform may differ from the film's premiere date.

The skill removes the film title and answer date while retaining the distinction between platform release and premiere.

All three cases are from the first bank update. WebShop and SearchQA use Qwen3-4B; ALFWorld uses Qwen3-8B. The WebShop group has no successful rollouts, and the SearchQA group has eight.

D SKILL-BANK DYNAMICS

We examine skill-bank dynamics for Qwen3-4B on ALFWorld, WebShop, and SearchQA over the first 150 training steps. Each task contributes 16 analyzed groups per step. Table 6 summarizes the resulting records, and Figure 4 shows bank size and rollout-group composition over this common window.

Table 6: Skill-bank statistics for Qwen3-4B over 150 training steps. Accepted group lessons pass the output-format checks.

	ALFWorld	WebShop	SearchQA
Bank snapshots	150	150	150
Analyzed groups	2,400	2,400	2,400
Accepted group lessons	2,031	2,381	2,383
Mixed-outcome groups	877	817	534
Minimum bank tokens	335	477	534
Maximum bank tokens	1,024	1,024	1,024

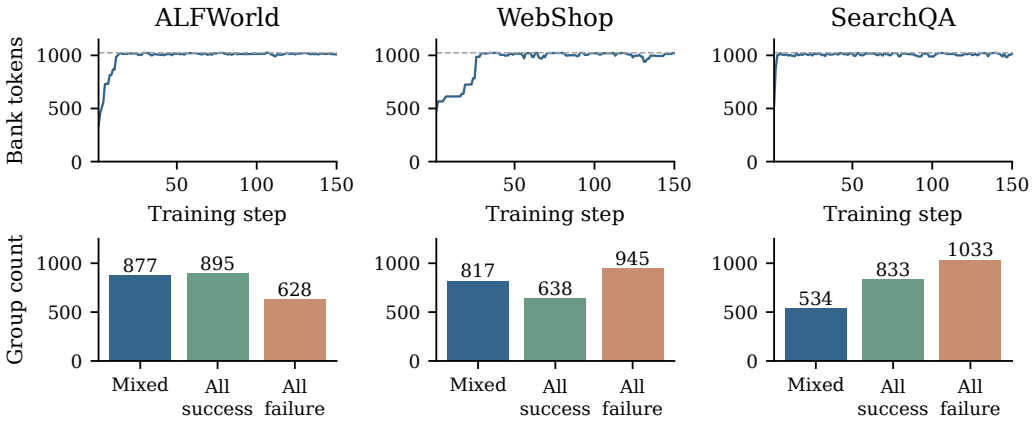

Figure 4: Skill-bank dynamics for Qwen3-4B across three tasks over 150 training steps. Top: bank token count at each step, without smoothing; the dashed line marks the 1,024-token budget. Bottom: total numbers of mixed-outcome, all-success, and all-failure groups analyzed in the same window.

Bank size. All three banks reach the 1,024-token budget. Subsequent token counts remain bounded while bank updates can replace existing content; token occupancy alone does not measure the diversity or correctness of retained skills.

Group composition. Mixed-outcome groups account for 36.5% of analyzed groups on ALFWorld, 34.0% on WebShop, and 22.3% on SearchQA. The analyzer therefore draws on both success–failure contrasts and groups with uniform outcomes across all three tasks.

E Detailed Results with Variability

Tables 7 and 8 report mean ± standard deviation across the three runs. Within each model scale, the best mean is bold with light red shading, and the second-best distinct mean is underlined with light blue shading.

Table 7: ALFWorld and WebShop results across three model scales (mean ± standard deviation).

	ALFWorld							WebShop	
Method	**Pick**	**Look**	**Clean**	**Heat**	**Cool**	**Pick2**	**Avg**	**Score**	**Succ.**
Qwen3-1.7B									
Vanilla	21.0 ± 6.7	30.8 ± 6.3	3.7 ± 3.0	2.1 ± 2.9	4.0 ± 3.3	5.6 ± 2.0	10.7 ± 1.0	44.26 ± 0.9	4.40 ± 0.8
GRPO	48.6 ± 2.3	38.5 ± 0.0	29.6 ± 3.0	41.7 ± 2.9	32.0 ± 0.0	20.8 ± 0.0	35.5 ± 0.3	63.03 ± 2.1	48.60 ± 1.8
OPSD	13.3 ± 1.3	30.8 ± 0.0	0.0 ± 0.0	6.2 ± 0.0	0.0 ± 0.0	4.2 ± 0.0	7.6 ± 0.3	43.2 ± 0.5	4.2 ± 0.8
GRPO+OPSD	54.3 ± 0.0	46.2 ± 0.0	42.0 ± 1.7	29.2 ± 2.9	40.0 ± 0.0	20.8 ± 0.0	40.0 ± 0.6	72.0 ± 0.2	48.9 ± 1.6
RLSD	52.4 ± 1.3	30.8 ± 0.0	50.6 ± 1.7	37.5 ± 0.0	**48.0 ± 3.3**	9.7 ± 2.0	40.2 ± 0.3	67.19 ± 0.8	48.13 ± 0.5
SDAR	**59.0 ± 1.3**	53.8 ± 0.0	46.9 ± 1.7	**56.2 ± 0.0**	36.0 ± 3.3	16.7 ± 0.0	44.5 ± 1.2	68.61 ± 0.4	52.00 ± 0.8
PLUME (ours)	**59.0 ± 2.7**	**56.4 ± 3.6**	**56.8 ± 1.7**	54.2 ± 2.9	40.0 ± 3.3	**48.6 ± 8.6**	**52.62 ± 0.9**	**74.60 ± 0.3**	**56.40 ± 0.0**
Qwen3-4B									
Vanilla	59.0 ± 2.7	30.8 ± 6.3	18.5 ± 3.0	22.9 ± 5.9	6.7 ± 1.9	22.2 ± 2.0	28.81 ± 1.2	11.98 ± 0.8	1.00 ± 0.2
GRPO	89.5 ± 1.3	**64.1 ± 9.6**	69.1 ± 4.6	66.7 ± 7.8	57.3 ± 3.8	48.6 ± 2.0	67.86 ± 1.2	75.03 ± 0.7	62.33 ± 0.3
OPSD	58.1 ± 3.6	28.2 ± 3.6	16.0 ± 4.6	31.2 ± 0.0	5.3 ± 1.9	12.5 ± 0.0	26.90 ± 1.4	10.6 ± 1.2	1.5 ± 0.2
GRPO+OPSD	**90.5 ± 1.3**	51.3 ± 9.6	59.3 ± 5.2	70.8 ± 2.9	45.3 ± 1.9	23.6 ± 2.0	59.0 ± 1.3	80.8 ± 0.1	69.1 ± 0.7
RLSD	83.8 ± 1.3	59.0 ± 3.6	76.5 ± 1.7	**83.3 ± 2.9**	58.7 ± 8.2	50.0 ± 0.0	69.8 ± 1.5	82.24 ± 0.7	70.80 ± 1.2
SDAR	77.1 ± 4.0	56.4 ± 3.6	72.8 ± 1.7	66.7 ± 2.9	57.3 ± 6.8	50.0 ± 5.9	65.0 ± 3.1	78.25 ± 1.6	66.80 ± 0.8
PLUME (ours)	86.67 ± 1.4	51.28 ± 9.6	**81.48 ± 5.2**	81.25 ± 5.1	**73.33 ± 1.9**	**52.78 ± 2.0**	**73.57 ± 1.0**	**85.51 ± 0.2**	**72.93 ± 0.4**
Qwen3-8B									
Vanilla	61.9 ± 1.4	33.3 ± 7.3	23.5 ± 3.5	20.8 ± 5.9	25.3 ± 5.0	33.3 ± 5.9	35.71 ± 0.0	26.53 ± 1.9	4.33 ± 1.1
GRPO	84.8 ± 1.3	64.1 ± 3.6	66.7 ± 3.0	62.5 ± 0.0	58.7 ± 7.5	52.8 ± 2.0	66.7 ± 2.0	84.55 ± 0.7	71.53 ± 1.1
OPSD	64.8 ± 1.3	35.9 ± 3.6	17.3 ± 7.6	20.8 ± 7.8	29.3 ± 1.9	30.6 ± 3.9	35.7 ± 0.6	12.6 ± 0.6	0.9 ± 0.2
GRPO+OPSD	83.8 ± 3.6	53.8 ± 0.0	74.1 ± 3.0	66.7 ± 5.9	66.7 ± 1.9	68.1 ± 2.0	71.4 ± 1.5	82.1 ± 0.9	71.4 ± 0.9
RLSD	88.6 ± 4.0	**66.7 ± 3.6**	67.9 ± 6.3	79.2 ± 2.9	58.7 ± 3.8	45.8 ± 3.4	68.81 ± 2.9	82.45 ± 0.8	71.00 ± 0.7
SDAR	82.9 ± 0.0	56.4 ± 3.6	84.0 ± 1.7	58.3 ± 2.9	**80.0 ± 5.7**	61.1 ± 2.0	73.6 ± 1.2	81.30 ± 0.7	71.87 ± 0.7
PLUME (ours)	**98.1 ± 1.3**	64.1 ± 3.6	**95.1 ± 1.7**	**87.5 ± 5.1**	78.7 ± 5.0	**73.6 ± 3.9**	**85.48 ± 1.5**	**86.51 ± 1.0**	**79.67 ± 0.9**

Table 8: SearchQA results across three model scales (mean ± standard deviation), including the seven QA datasets and the ID, OOD, and overall aggregates.

	QA datasets							Aggregates		
Method	**NQ**	**Triv**	**Pop**	**Hotp**	**2Wk**	**MuS**	**Bam**	**ID**	**OOD**	**Avg**
Qwen3-1.7B										
Vanilla	22.15 ± 0.6	33.39 ± 0.6	21.71 ± 1.0	17.88 ± 0.7	18.35 ± 0.3	3.94 ± 0.4	14.13 ± 1.6	19.28 ± 0.6	22.84 ± 0.5	22.08 ± 0.5
GRPO	40.65 ± 0.3	57.59 ± 0.2	43.96 ± 0.3	35.20 ± 0.2	30.60 ± 0.2	11.47 ± 0.3	29.07 ± 0.4	36.99 ± 0.1	41.64 ± 0.1	40.65 ± 0.0
OPSD	6.42 ± 0.3	15.62 ± 0.6	10.46 ± 0.4	8.01 ± 0.3	16.34 ± 0.2	1.24 ± 0.2	9.07 ± 1.4	7.49 ± 0.3	13.16 ± 0.3	11.95 ± 0.3
GRPO+OPSD	40.95 ± 0.0	**58.73 ± 0.1**	42.91 ± 0.1	**37.63 ± 0.0**	33.89 ± 0.1	12.80 ± 0.3	30.67 ± 1.0	**38.72 ± 0.0**	42.69 ± 0.1	41.85 ± 0.1
RLSD	38.59 ± 0.2	56.40 ± 0.0	41.56 ± 0.0	35.79 ± 0.2	34.13 ± 0.3	12.94 ± 0.1	**35.47 ± 0.8**	36.71 ± 0.2	41.67 ± 0.1	40.61 ± 0.1
SDAR	39.31 ± 0.2	57.31 ± 0.0	43.66 ± 0.2	36.53 ± 0.2	36.68 ± 0.2	**13.25 ± 0.3**	30.67 ± 0.8	37.44 ± 0.2	43.45 ± 0.1	42.17 ± 0.1
PLUME (ours)	**41.14 ± 0.4**	58.04 ± 0.1	**44.66 ± 0.1**	35.80 ± 0.4	**37.08 ± 0.3**	11.41 ± 0.2	30.13 ± 0.9	37.55 ± 0.4	**44.02 ± 0.1**	**42.64 ± 0.1**
Qwen3-4B										
Vanilla	30.09 ± 0.6	49.37 ± 0.3	31.11 ± 0.7	25.90 ± 0.3	28.89 ± 0.2	7.10 ± 0.4	28.53 ± 1.0	27.27 ± 0.4	34.07 ± 0.4	32.62 ± 0.4
GRPO	47.05 ± 0.3	**64.03 ± 0.1**	45.60 ± 0.1	40.99 ± 0.1	42.70 ± 0.3	15.38 ± 0.2	37.07 ± 0.4	42.98 ± 0.1	48.01 ± 0.1	46.94 ± 0.1
OPSD	3.32 ± 2.7	8.73 ± 5.7	4.44 ± 3.0	2.60 ± 1.6	3.90 ± 2.1	0.46 ± 0.3	2.93 ± 3.0	2.84 ± 1.9	5.23 ± 3.2	4.72 ± 2.9
GRPO+OPSD	**47.53 ± 0.2**	63.81 ± 0.2	**47.54 ± 0.2**	42.75 ± 0.1	42.28 ± 0.1	16.94 ± 0.2	42.40 ± 0.7	44.32 ± 0.1	**48.60 ± 0.1**	**47.69 ± 0.1**
RLSD	45.74 ± 0.3	62.56 ± 0.1	44.66 ± 0.1	40.95 ± 0.2	37.54 ± 0.1	**18.18 ± 0.3**	**45.07 ± 1.4**	42.52 ± 0.1	45.87 ± 0.1	45.15 ± 0.1
SDAR	45.84 ± 0.1	61.08 ± 0.1	43.06 ± 0.1	40.21 ± 0.0	40.07 ± 0.1	17.60 ± 0.3	42.13 ± 1.6	42.05 ± 0.0	45.63 ± 0.0	44.87 ± 0.0
PLUME (ours)	47.18 ± 0.1	63.15 ± 0.1	45.34 ± 0.3	**43.15 ± 0.2**	**44.18 ± 0.1**	18.05 ± 0.4	43.20 ± 0.8	**44.47 ± 0.2**	48.31 ± 0.1	47.49 ± 0.0
Qwen3-8B										
Vanilla	27.03 ± 0.5	53.01 ± 0.4	28.26 ± 0.2	26.73 ± 0.3	31.36 ± 0.2	7.61 ± 0.4	34.93 ± 1.0	26.82 ± 0.4	34.89 ± 0.2	33.17 ± 0.2
GRPO	46.90 ± 0.2	65.42 ± 0.3	44.08 ± 0.1	44.29 ± 0.1	47.66 ± 0.2	19.68 ± 0.2	**48.27 ± 0.4**	45.15 ± 0.1	49.68 ± 0.1	48.71 ± 0.1
OPSD	31.03 ± 1.6	49.25 ± 1.4	36.22 ± 0.6	28.43 ± 1.1	29.28 ± 2.4	11.16 ± 0.5	34.40 ± 1.3	29.28 ± 1.0	36.20 ± 0.8	34.73 ± 0.9
GRPO+OPSD	48.01 ± 0.1	**65.74 ± 0.1**	47.42 ± 0.1	44.29 ± 0.2	44.86 ± 0.1	19.67 ± 0.1	46.67 ± 1.4	45.50 ± 0.1	50.07 ± 0.0	49.10 ± 0.0
RLSD	47.54 ± 0.3	64.49 ± 0.2	46.14 ± 0.1	44.96 ± 0.2	42.74 ± 0.1	20.22 ± 0.3	46.13 ± 0.4	45.80 ± 0.2	48.65 ± 0.1	48.04 ± 0.0
SDAR	47.90 ± 0.2	64.49 ± 0.2	45.93 ± 0.1	43.11 ± 0.1	42.99 ± 0.2	19.18 ± 0.3	45.87 ± 1.0	44.68 ± 0.1	48.59 ± 0.1	47.76 ± 0.1
PLUME (ours)	**49.50 ± 0.1**	65.24 ± 0.1	**47.70 ± 0.1**	**45.47 ± 0.0**	**48.06 ± 0.1**	**21.36 ± 0.3**	46.93 ± 1.2	**46.81 ± 0.1**	**51.12 ± 0.0**	**50.20 ± 0.0**

F Policy Generalization on ALFWorld Unseen Environments

We further evaluate whether PLUME's policy improvements extend to unseen environments in ALFWorld, complementing the skill-bank transfer experiment in Table 3. We evaluate GRPO and PLUME at three model scales on all 134 tasks in the `valid_unseen` split and report the mean and population standard deviation of task success rates across three runs.

Table 9: Policy performance on the ALFWorld unseen split. Success rates are reported as mean ± standard deviation over three runs. Δ denotes the improvement of PLUME over GRPO in percentage points.

Model	GRPO (%)	PLUME (%)	Δ (pp)
Qwen3-1.7B	29.10 ± 0.61	**43.78** ± 3.96	+14.68
Qwen3-4B	56.47 ± 1.86	**74.13** ± 2.46	+17.66
Qwen3-8B	61.19 ± 2.11	**79.60** ± 1.86	+18.41

Results. As shown in Table 9, PLUME outperforms GRPO at all three model scales, improving success rates by 14.68, 17.66, and 18.41 percentage points for Qwen3-1.7B, Qwen3-4B, and Qwen3-8B, respectively. These gains show that PLUME's policy improvements extend to the benchmark's unseen environments.