EVOLVING ROLLOUTS: Harnessing Historical Experience for Web Agent Evolution in Reinforcement Learning

Anonymous Authors[1]

Abstract

Agentic reinforcement learning (RL) for web search is prohibitively expensive due to long context lengths and costly environment interactions, and this inefficiency is further exacerbated by GRPO-based optimization, which discards learning signals from entire rollout groups with zero reward variance. In this work, we propose EVOLVING ROLLOUTS, an RL framework for web-search agents that moves beyond episodic training and distills collected rollouts into in-context guidance for future policy behavior. By extracting the reward-labeled trajectories into strategic experiences, our method augments standard parameter-space optimization with implicit context-space optimization guided by prior experience. This enables the agent to recover learning signals from zero-variance rollouts, thereby fostering co-evolution between the policy and the experience repository. EVOLVING ROLLOUTS improves sample efficiency and task performance across representative web search benchmarks, enabling Qwen3-4B models to achieve performance comparable to that of the substantially larger Qwen3-30B-A3B model on GAIA, Xbench, and HLE. We open-source our training framework to support reproducibility and future research.[1]

1. Introduction

Large Reasoning Models (LRMs) have recently demonstrated strong capabilities on complex reasoning tasks, driven in part by reinforcement learning from verifiable reasoning (RLVR) (Guo et al., 2025; Achiam et al., 2023; Yang et al., 2025a). The increasing complexity of reasoning tasks necessitates access to external knowledge sources, iterative planning, and tool interaction, leading to the use of these models as autonomous agents for multi-hop, tool-augmented tasks such as web search and deep research (Zheng et al., 2025; Yao et al., 2026; Wu et al., 2025b; Li et al., 2025c; Qin et al., 2025; Zhu et al., 2025; Shi et al., 2025; Zhou et al., 2023). This shift from static reasoning to agentic decision making introduces substantial challenges. Agentic tasks unfold over long horizons and require sustained interaction with dynamic web environments (Li et al., 2025c; Liu et al., 2025; Li et al., 2025b; Wang et al., 2025a; Liang et al., 2025), often incurring high costs from API calls, web crawling, and other external tool execution. Reinforcement learning offers a principled framework for optimizing such behaviors under delayed rewards, yet remains highly inefficient in practice: episodes are long, rewards are sparse, and policy updates critically depend on reward variance across parallel rollouts. Current Agentic web search RL (Tao et al., 2025; Wu et al., 2025a; Li et al., 2025a) predominantly relies on binary, outcome-based rewards, which results in low sample efficiency and unstable training (Yu et al., 2025b). Specifically, when all rollouts within a group either succeed or fail, reward normalization causes the policy gradient to vanish, resulting in the removal of entire batches of costly environment interactions, even though these trajectories may contain partial reasoning processes and informative failure signals. Process reward models (PRMs) partially alleviate this issue by providing denser supervision, but they rely on costly step-level annotation (Chae et al., 2025; Cui et al., 2025) and remain largely static, making them poorly suited to adapt to new reasoning patterns that emerge during reinforcement learning. These limitations highlight the need for learning paradigms that can extract reusable learning signals from zero-reward-variance rollouts without relying on additional annotations or fixed reward structures.

To address this limitation, we draw inspiration from recent experience-based agent learning paradigms, which emphasize treating failures as valuable sources of reusable experience (Li et al., 2025c; Chhikara et al., 2025; Wang et al., 2024; Fang et al., 2025; Zhang et al., 2025; Tang et al., 2025; Wang et al., 2025b; Yu et al., 2025a). These approaches suggest that improving agentic reasoning by reusing historical

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1]Code is provided in the supplementary materials.

interactions can indirectly alleviate optimization difficulties, especially when reward signals are sparse. However, existing methods predominantly exploit experience at test time (Cai et al., 2025; Ouyang et al., 2025; Wei et al., 2025; Yang et al., 2025b) or treat experience repositories as static artifacts, without tightly integrating experience reuse into the reinforcement learning process itself.

Building on this insight, we propose EVOLVING ROLLOUTS, a reinforcement learning framework that jointly optimizes the agent policy and a co-evolving experience repository. Rather than discarding rollout groups with uniform rewards, EVOLVING ROLLOUTS distills reward-labeled trajectories into a structured and continually updated experience bank, which is reused as in-context guidance for subsequent episodes. By unifying parameter-space optimization with context-space experience evolution, EVOLVING ROLLOUTS recovers learning signals from otherwise wasted interactions and substantially improves sample efficiency under constrained training budgets. Empirically, EVOLVING ROLLOUTS improves both optimization stability and final performance across representative web-agent benchmarks. Under the same training budget, EVOLVING ROLLOUTS enables a Qwen3-4B model to achieve 44.7% on GAIA, a +3.0% improvement over the GRPO baseline, matching the performance of the significantly larger Qwen3-30B-A3B model.

In summary, our main contributions are as follows:

- **Inefficiency in agent RL:** Zero-variance rollout groups yield degenerate gradients and hinder optimization, with a stronger effect in small-scale models.
- **Co-evolving training framework:** Policy optimization is tightly integrated with a structured experience repository that is continuously distilled and reused.
- **Improved stability and sample efficiency:** Extensive experiments show more stable training and stronger performance under same interaction budgets.
- **Open-source framework:** We release our training framework to support reproducibility and future research.

2. Related Work

2.1. LLM-based Web Search Agents

Recent LLM-based web search agents have evolved from static retrieval to long-horizon information-seeking with autonomous planning and tool use. Systems such as WebResearcher (Qiao et al., 2025) and WebSailor-V2 (Li et al., 2025a) leverage iterative planning and reinforcement learning to operate under dynamic, partially observable web environments. To address limited supervision, prior work emphasizes scalable data construction and training, including formalization-driven task synthesis in WebShaper (Tao et al., 2025) and curriculum-based learning in WebDancer (Wu et al., 2025a). Complementary architectural designs, such as WebWalker's Explorer–Critic framework (Wu et al., 2025c) and multimodal WebWatcher (Geng et al., 2025), further improve robustness during environment interaction. Despite these advances, most work focuses on task coverage, exploration, or data scale, while the sample efficiency of reinforcement learning remains underexplored. In particular, GRPO often drops entire rollout groups when reward variance is zero, wasting expensive interaction trajectories—including failed or partially correct attempts that still carry useful reasoning signals. This motivates training paradigms that better reuse collected rollouts and extract supervision beyond standard reward-based policy optimization.

2.2. Experience-driven Agentic Evolution

Limitations of task-coupled memory have spurred experience-driven agent evolution, where agents distill and reuse generalizable rules from past trajectories. Early Experience (Zhang et al., 2025) learns from reward-free self-generated interactions, while AgentKB (Tang et al., 2025) externalizes reusable patterns via in-context retrieval for cross-domain transfer. This trend emphasizes test-time learning, benchmarked by Evo-Memory (Wei et al., 2025), and instantiated by MUSE's Plan–Execute–Reflect–Memorize loop (Yang et al., 2025b) and FLEX's evolvable in-context experience library (Cai et al., 2025). Recent work (ReasoningBank (Ouyang et al., 2025), AgentEvolver (Zhai et al., 2025)) further distills transferable strategies and failure lessons for autonomous improvement. However, these methods decouple context-space experience evolution from parameter-space RL, and seldom co-optimize experience extraction and experience reuse, motivating unified frameworks where experience evolution and RL updates co-evolve in a single loop.

3. Method

We present EVOLVING ROLLOUTS, a reinforcement learning framework that transforms costly web-agent rollouts into reusable and evolving strategic knowledge. Our key insight is to complement standard parameter-space optimization with a persistent context-space optimization channel, enabling experience reuse even when gradient signals vanish. Our approach operates in three stages: (1) *cold-start experience construction*, where we distill high-quality trajectories from open-source datasets into an initial experience repository; (2) *experience-driven workflow execution and supervised fine-tuning (SFT)*, where the agent retrieves relevant experiences during inference and is fine-tuned on self-generated, context-rich trajectories; and (3) *group-based RL with evolving memory*, where we apply Group Relative Policy Optimization (GRPO) on new tasks and continuously evolve the experience repository through contrastive extrac-

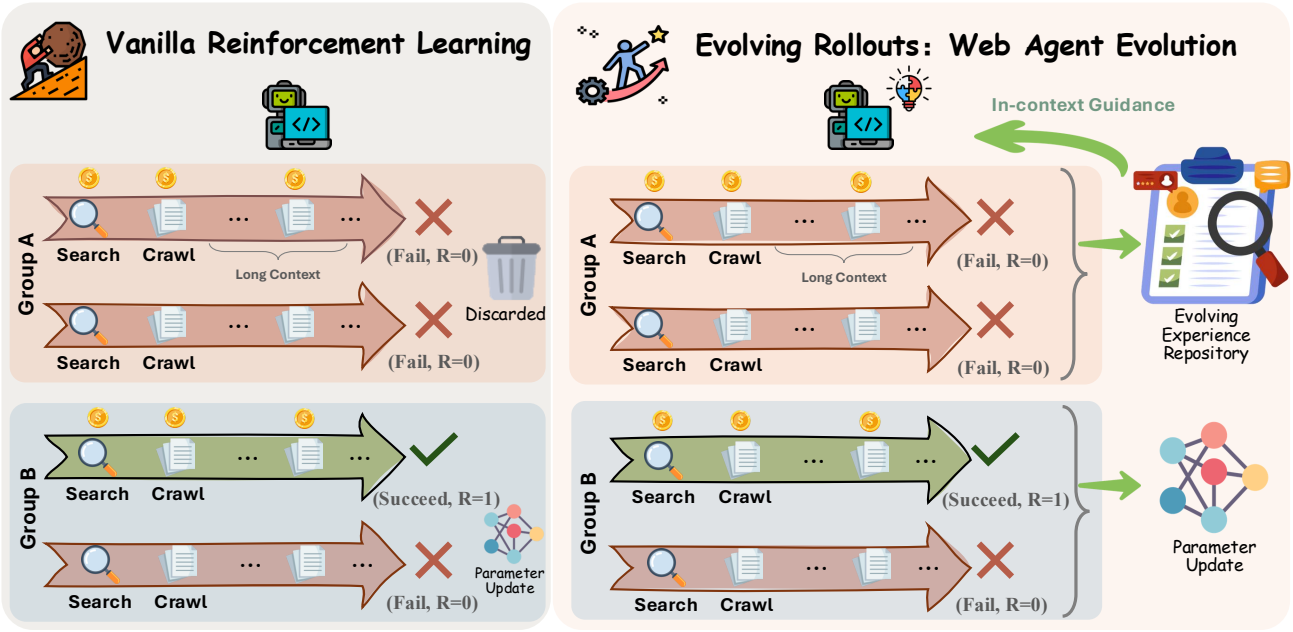

Figure 1. The framework of EVOLVING ROLLOUTS. Naive GRPO-based reinforcement learning updates parameters only from rollout groups with non-zero reward variance (Group B), discarding signals from all costly trajectories from zero reward variance group (Group A). EVOLVING ROLLOUTS recovers the signals from zero reward variance rollout into the experience repository context space, which provides in-context guidance for future rollouts while preserving explicit parameter space optimization.

tion, experience consolidation, and experience updating. By co-evolving the policy and its memory, our method ensures that no successful interaction is wasted, turning every rollout into a stepping stone for future improvement.

3.1. Problem Formulation

We consider a web agent tasked with answering a user query q through sequential interactions with a web environment. The agent is instantiated as a Large Reasoning Model (LRM) π_θ that, at each time step t, observes an environment observation o_t (e.g., webpage content) and selects an action from a discrete action space. The available actions include: SEARCH(k), which issues a new search query k; CRAWL(p), which extracts relevant information following an instruction p; RETRIEVE(q), which queries the experience repository $\mathscr{E}$ to obtain previously distilled strategies relevant to the current query q; and ANSWER(y), which outputs a final answer y and terminates the episode.

An interaction episode induces a trajectory $\tau = (q, o_1, a_1, \ldots, o_T, a_T)$ drawn from the policy-induced distribution $\tau \sim \pi_\theta(\cdot \mid q)$. Each trajectory terminates with an ANSWER action and is assigned a scalar reward $r \in [0,1]$ that reflects the degree of task success.

To support experience reuse, we maintain an experience repository $\mathscr{E} = \{e_1, e_2, \ldots, e_n\}$ that stores distilled, transferable reasoning strategies. Each experience e_i is represented as a structured tuple:

$$e_i = (\text{title}_i, \text{desc}_i, \text{content}_i, \mathbf{v}_i \in \mathbb{R}^d, \text{success}_i) \quad (1)$$

where title_i denotes a concise strategy identifier, desc_i provides a one-sentence summary, content_i contains detailed and transferable reasoning content, $\mathbf{v}_i$ is a d-dimensional semantic embedding, and success_i is a binary episode-level indicator of task completion. The semantic embeddings enable efficient similarity-based retrieval and consolidation of experiences within the repository.

3.2. Initial Experience Repository Construction

To bootstrap the experience repository $\mathscr{E}$, we leverage publicly available web-agent trajectory datasets [2] as cold-start data. Each trajectory in these datasets is a reward-labeled rollout generated by a base agent in a web environment. We process each trajectory using a frozen Large Language Model (LLM) to extract a single, self-contained experience that distills the core reasoning strategy and action pattern leading to task success. Specifically, the LLM reformats the raw observation–action sequence into a natural language prompt–response pair suitable for supervised fine-tuning, effectively distilling each trajectory into a reusable, strategy-level learning signal. Implementation details of

[2] https://huggingface.co/datasets/PersonalAILab/AFM-MHQA-Agent-SFT-Dataset

the extraction prompts and formatting rules are provided in Appendix A. The resulting set of distilled experiences constitutes our initial repository $\mathscr{E}$, which serves as the training data for the subsequent SFT phase.

Formally, given a trajectory $\tau = (q, o_1, a_1, \ldots, o_T, a_T)$, the extraction yields an experience instance:

$$e = \textsc{Extract}_{\phi}\left(q, \{o_t, a_t\}_{t=1}^{T}\right) \tag{2}$$

where $\textsc{Extract}_{\phi}$ denotes a frozen LLM-based extractor parameterized by ϕ. Collecting such experiences from a set of trajectories $\{\tau^{(i)}\}_{i=1}^{N}$, we construct the initial experience repository as:

$$\mathscr{E} = \left\{e^{(i)}\right\}_{i=1}^{N}. \tag{3}$$

3.3. Experience-Driven Workflow and Supervised Fine-Tuning

We design a structured web-search workflow in which a base Large Reasoning Model (LRM) π_{θ_0} autonomously executes tasks by interleaving reasoning, tool invocation (SEARCH, SCRAPE), experience retrieval (RETRIEVE), and final answer generation. At execution time, the agent may invoke $\textsc{Retrieve}(q)$ to fetch relevant past experiences from the experience repository $\mathscr{E}$, which are injected as in-context strategic guidance to condition subsequent reasoning and tool-use decisions. This process yields a set of execution trajectories $\mathscr{T} = \{\tau^{(i)}\}_{i=1}^{N}$, where each trajectory $\tau^{(i)} = (q^{(i)}, h^{(i)}, y^{(i)})$ consists of a user query $q^{(i)}$, the full interaction history $h^{(i)}$ (including retrieved experiences, if any), and the final output $y^{(i)}$. From $\mathscr{T}$, we construct an experience-conditioned SFT dataset by treating each trajectory as a prompt–target pair. The SFT objective is given by:

$$\mathscr{L}_{\text{SFT}} = \mathbb{E}_{(q,h,y)\sim\mathscr{T}}\left[-\log P_{\theta}(y \mid q, h)\right], \tag{4}$$

which yields an improved policy π_{θ} that internalizes successful behaviours from the generated experience trajectories.

3.4. Evolving Reinforcement Learning

Following SFT, we employ Group Relative Policy Optimization (GRPO), wherein the policy and experience repository $\mathscr{E}$ co-evolve each training iteration refines model parameters while simultaneously updating the strategic knowledge base. Standard GRPO updates policy parameters using group-relative advantages. For a prompt x, we sample K rollouts $\mathscr{G} = \{\tau_1, \ldots, \tau_K\}$ and normalize rewards within the group:

$$\hat{A}(\tau) = \frac{r(\tau) - \mu_{\mathscr{G}}}{\sigma_{\mathscr{G}} + \varepsilon} \tag{5}$$

The policy gradient then becomes:

$$\nabla_{\theta}\mathscr{L}_{\text{GRPO}} = \mathbb{E}_{\tau\sim\mathscr{G}}\left[\nabla_{\theta}\log\pi_{\theta}(\tau)\cdot\hat{A}(\tau)\right] \tag{6}$$

However, this *parameter-space optimization* yields zero gradient when all rollouts share the same outcome ($\sigma_{\mathscr{G}} = 0$), eliminating learning signals despite high rollout cost. In web agent settings, where each rollout incurs substantial computational and API cost, discarding these groups leads to significant inefficiency. Therefore, we introduce a complementary *context-space optimization* mechanism that operates by evolving the experience repository $\mathscr{E}$, forming an auxiliary learning channel independent of gradient variance. Since the agent retrieves from $\mathscr{E}$ during inference, an improved repository directly enhances policy behavior through in-context guidance, without modifying θ. Critically, this context-space channel operates on *all* rollouts regardless of reward variance, ensuring no expensive trajectory is wasted.

3.4.1. EXPERIENCE EXTRACTION.

At each training step, GRPO generates rollout groups $G_q = \{(\tau_1, r_1), \ldots, (\tau_m, r_m)\}$, where m trajectories attempt the same question q. We distill generalizable experience through contrastive analysis across these parallel attempts.

We first compress each trajectory into step-level intended action summaries, distilling the agent's decision rationale at each step:

$$\tau_j^{\text{comp}} = \textsc{Compress}_{\phi}(\tau_j) = \{(s_t, a_t^{\text{intent}})\}_{t=1}^{T_j} \tag{7}$$

where a_t^{intent} captures the high-level goal behind each action. We then construct a contrastive prompt presenting all compressed trajectories with their binary outcome labels $\ell_j \in \{0, 1\}$ indicating failure or success, and extract transferable experience as:

$$E_{\text{new}} = \textsc{Extract}_{\phi}\left(q, \{(\tau_j^{\text{comp}}, \ell_j)\}_{j=1}^{m}\right) \tag{8}$$

In contrast to gradient-based optimization, this experience extraction procedure applies to any reward configuration. All success groups yield distilled winning strategies, all failure groups yield anti patterns, and mixed groups enable direct comparison between effective and ineffective approaches. As a result, rollout groups with zero reward variance, which are typically discarded in standard parameter space optimization, are transformed into valuable experience contributions.

3.4.2. EXPERIENCE CONSOLIDATION.

To control the growth of the experience repository and reduce redundancy, we perform similarity-based consolidation among stored experiences. Let $e^* = \arg\max_{e\in\mathscr{E}} \text{sim}(\mathbf{v}_{e_{\text{new}}}, \mathbf{v}_e)$ denote the most semantically similar existing experience to a newly extracted experience e_{new}, where $\mathbf{v}_e$ represents the embedding of experience e. When incorporating e_{new} into the repository, the update rule is

defined as:

$$\mathscr{E} \leftarrow \mathscr{E} \cup \{e_{\text{new}}\} \setminus \{e^* \mid \text{sim}(\mathbf{v}_{e_{\text{new}}}, \mathbf{v}_{e^*}) > \theta\} \quad (9)$$

This consolidation strategy preferentially retains more recent, policy-aligned experiences under high semantic overlap. Because newly extracted experiences capture strategies induced by the current policy state, this replacement mechanism ensures that the experience repository remains compact while continuously evolving its stored knowledge in alignment with the evolving policy throughout training.

4. Experiments

4.1. Experimental Setup

4.1.1. BENCHMARKS.

We evaluate our method on three representative web-agent benchmarks: **GAIA**[1] (Mialon et al., 2023), **xBench-DeepSearch** (Xbench-Team, 2025), and **HLE**[2] (Phan et al., 2025). These benchmarks collectively cover a wide spectrum of agentic challenges, including multi-hop web search, long-horizon reasoning, and tool invocation.

4.1.2. MODELS AND TRAINING VARIANTS.

We conduct experiments using **Qwen-3** (Qwen et al., 2025) backbone at the **4B** scale. The model is integrated into an identical **ReAct**-style agent loop for action selection and tool invocation, ensuring that performance differences arise solely from training dynamics and experience mechanisms. This model size is chosen to reflect practical deployment regimes for web agents in resource-constrained and edge-oriented settings.

For fair comparison, we evaluate a consistent set of training variants: (i) the base instruction-tuned model without post-training (**ReAct**), (ii) supervised fine-tuning (**SFT**) on curated agent trajectories, (iii) reinforcement learning using **GRPO** (Shao et al., 2024) on top of SFT (**RL**), and (iv) our proposed method, EVOLVING ROLLOUTS. EVOLVING ROLLOUTS augments GRPO with rollout recycling and an evolving experience component that provides structured in-context guidance during training. Importantly, EVOLVING ROLLOUTS does not modify the underlying policy optimization objective or reward function, isolating the effect of historical rollout reuse.

Additionally, we compare with a strong reference model, **Qwen3-30B-A3B**, to contextualize the performance of our 4B model against a significantly larger baseline.

[1]GAIA: 103 text-only instances (Li et al., 2025c).

[2]HLE: 100 instances from the text-only split.

Table 1. Main results on GAIA, xBench-DeepSearch, and HLE benchmarks (Pass@1). EVOLVING ROLLOUTS with evolving experience bank achieves the best average performance (31.2), matching the 30B reference model on GAIA and HLE while using only 4B parameters.

Method	GAIA	xBench	HLE	Avg.
Reference Model				
Qwen3-30B-A3B	44.7	39.0	13.0	32.2
Naive Training Pipeline				
ReAct	24.3	34.0	10.0	22.8
+ SFT	34.0	34.0	8.0	25.3
+ SFT + RL	41.7	35.0	10.0	28.9
EVOLVING ROLLOUTS (Ours)				
ReAct (w/ Experience)	30.1	32.0	7.0	23.0
+ SFT	35.9	32.0	11.0	26.3
+ SFT + RL (static)	42.7	**38.0**	11.0	30.6
+ SFT + RL (evolving)	**44.7**	36.0	**13.0**	**31.2**

4.1.3. EVALUATION METRIC.

In the main comparison, we report **Pass@1**, defined as the fraction of evaluation instances for which the agent successfully completes the task in a single rollout. Pass@1 captures single-trajectory success without relying on multiple sampling attempts. We intentionally focus on Pass@1 to reflect realistic deployment scenarios where only one trajectory is executed per query.

4.2. Main Results

Table 1 reports the performance across all evaluated benchmarks. EVOLVING ROLLOUTS demonstrates consistent and robust performance gains across all training stages.

4.2.1. IMPROVEMENTS OVER RL BASELINES.

Relative to the SFT+RL baseline without experience (28.9 avg), EVOLVING ROLLOUTS achieves 2.3% improvement, reaching 31.2% on average. On GAIA, performance increases from 41.7% to 44.7%, representing a 3.0% absolute gain and matching the accuracy of the substantially larger Qwen3-30B-A3B reference model. Notably, the improvement from SFT to EVOLVING ROLLOUTS is larger than that obtained by adding RL alone, indicating that experience evolution provides benefits beyond standard RL.

4.2.2. EFFECTS OF EXPERIENCE WITH POLICY.

Across training stages, incorporating experience consistently improves performance, with the magnitude of improvement increasing as the base policy becomes stronger. For the base model, experience yields a modest gain from 22.8% to 23.0%, while for the SFT model it improves performance from 25.3% to 26.3%. The effect is most pronounced

in the RL setting, where experience raises the average score from 28.9% to 30.6%. This trend suggests that experience-conditioned guidance becomes increasingly effective as the policy's reasoning and action capabilities improve.

4.2.3. REDUCING THE GAP TO LARGER MODELS.

Despite its smaller scale, the 4B EVOLVING ROLLOUTS agent matches the 30B reference model on GAIA and HLE, and achieves 92% of its performance on xBench. These results indicate that evolving rollouts can substantially reduce the performance gap between smaller and larger models on challenging agentic benchmarks.

We provide a detailed analysis of the mechanisms underlying these gains in Section 5, including controlled ablations on policy–experience co-evolution, temporal pairing diagnostics, and experience quality.

5. Ablation Studies

This section investigates the mechanisms underlying EVOLVING ROLLOUTS by disentangling the contributions of policy optimization and experience evolution, and by characterizing how the experience repository evolves throughout training. Our ablations are designed to answer two key questions:(1) whether the performance gains arise from policy learning, experience evolution, or their interaction; and (2) how the composition and utilization of experience change as training progresses.

5.1. Decoupling Experience and Policy Contribution

We use two complementary diagnostics to decouple the roles of policy learning and experience evolution. First, Table 2 reports *training-time* ablations using three binary switches: whether training is **with experience** (**w/ Exp**), whether the experiences are updated over training (**Evolve Exp**), and whether the policy parameters are updated (**Evolve Policy**). This isolates the effects of (i) policy-only optimization without experience, (ii) policy optimization with *static experience*, (iii) evolving experience with a frozen policy, and (iv) co-evolving both (EVOLVING ROLLOUTS). Second, Figure 2 provides a *test-time* diagnostic on GAIA by swapping the pairing between checkpoints and experience snapshots (synced vs fixed-policy) without any additional training.

Table 2 shows that evolving only one component is insufficient. Training *with static experience* while evolving the policy improves over the no-experience baseline (30.6 vs 28.9 avg), but co-evolving both achieves the best overall performance (31.2 avg). In contrast, evolving experience while freezing the policy degrades performance sharply (23.0 avg), even below the no-experience baseline.

Figure 2 further decomposes the gains *at test time*. Keeping the policy fixed at the Step 5 checkpoint while swapping in later experience snapshots (orange) still yields an overall improvement, indicating that the evolved experience itself becomes more useful over training. However, the synced pairing (blue) is consistently higher, and the persistent gap (green shaded region) quantifies the additional gain attributable to policy evolution. Concretely, both curves share the same baseline at step 5 (35.9% GAIA), while the synced pairing exceeds the fixed-policy pairing by +5.8% at step 10 (39.8% vs 34.0%), +9.7% at step 20 (42.7% vs 33.0%), and +7.8% at step 31 (44.7% vs 36.9%).

The failure of the frozen-policy variant and the fixed-policy test-time pairing reveals a common principle: *experiences must remain aligned with the policy that consumes them*. In EVOLVING ROLLOUTS, experiences are extracted from contrastive rollout groups (e.g., all-fail vs all-correct), capturing the transition from failure to success that is meaningful for the *current* policy. As the policy improves, the evolving experience increasingly reflects strategies and failure modes discovered by later checkpoints. An early (Step 5) policy,

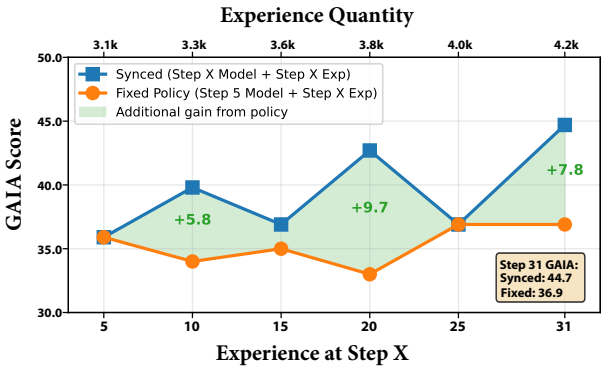

Figure 2. Test-time pairing decomposition on GAIA (all checkpoints and experience snapshots are taken from the same co-evolution run). Blue: synced pairing (Step X model + Step X experience snapshot). Orange: fixed-policy pairing (Step 5 model + Step X experience snapshot). The green shaded area represents the additional gain from policy evolution under matched pairing.

Table 2. Training-time ablations that isolate which components evolve. Rows (top to bottom) correspond to: (1) policy only (no experience), (2) policy + static experience, (3) evolving experience only (frozen policy), and (4) co-evolution (policy + experience). Adding a *static* experience repository improves performance (+1.7 avg), while evolving the repository without evolving the policy degrades performance (-5.9 avg). Co-evolving both achieves the best overall performance (31.2 avg).

w/ Exp	Evolve Exp	Evolve Policy	GAIA	xBench	HLE	Avg.
		✓	41.7	35.0	10.0	28.9
✓		✓	42.7	38.0	11.0	30.6
✓	✓		33.0	26.0	10.0	23.0
✓	✓	✓	**44.7**	**36.0**	**13.0**	**31.2**

however, may not reliably select or apply these newer strategies: it issues weaker queries, lacks the capability to execute more complex guidance, and faces a distribution shift between its own rollouts and the later-step contrastive patterns encoded in the evolving experience. As a result, evolving (or swapping in) later experiences without a correspondingly evolved policy can be irrelevant or confusing rather than helpful.

The contrast with EVOLVING ROLLOUTS is stark. With co-evolution training (and synced test-time pairing), the policy improves its ability to *use* experience while simultaneously generating higher-quality experience, forming a virtuous cycle in which better policies yield better experience, which yields better guidance, which further improves the policy. Overall, these results show that the benefit of EVOLVING ROLLOUTS arises from the *interaction* between policy and experience evolution, not from either component alone.

5.2. Experience Dynamics

To understand how experience evolves during training, we analyze its growth patterns and utilization statistics across training steps.

5.2.1. EXPERIENCE GROWTH AND COMPOSITION SHIFT.

The experience collection grows from 2,996 to 4,197 experiences over training, with an average of 44.9 changes per step (additions + merges). Notably, it exhibits a **stable churn rate of 1.0% to 2.1%**, indicating controlled evolution rather than volatile turnover.

More importantly, the experience composition shifts substantially over training, as visualized in Figure 3. At step 5, the experience is dominated by initialized experiences (2,942 SFT vs 230 RL, or 7% RL). By step 30, RL-generated experiences constitute **31%** of the experience set (1,274 of 4,162), well-integrated throughout the embedding space rather than forming isolated clusters. This gradual integration of RL experiences reflects the policy's growing capability to generate useful guidance: as the policy improves through RL, it produces higher-quality rollouts from which valuable experiences can be extracted. The initialized experiences provide a stable foundation of curated knowledge, while RL experiences capture emergent problem-solving strategies discovered during training.

Beyond source composition, the experience set maintains a balanced distribution between success-derived and failure-derived knowledge. As shown in Figure 4, the final experience set contains 2,162 success strategies (51.5%) and 2,035 failure lessons (48.5%), with the two types semantically interleaved rather than clustered separately.

5.2.2. DIVERSITY OF GENERATED EXPERIENCES.

A key question is whether the co-evolving policy generates increasingly diverse or repetitive experiences as training progresses. We analyze the embedding distribution of *newly generated* experiences at each training step, measuring spread (standard deviation), coverage (convex hull area), and internal diversity (average pairwise distance).

The results in Figure 5 reveal a striking pattern that underscores the benefits of co-evolution: newly generated experiences become **significantly more diverse** over training. Spread increases from 29.5 to 39.8 ($p = 0.01$), convex hull coverage expands from 59% to 74% of the embedding space ($p = 0.05$), and internal diversity grows from 36.8 to 49.6 ($p = 0.01$). Most notably, the distance from newly generated experiences to initial SFT experiences grows steadily from 0.91 to 1.51, confirming that the policy progressively discovers **genuinely novel problem-solving patterns** beyond its initial supervised training.

This diversity growth demonstrates three key advantages of co-evolution. First, the policy **avoids mode collapse**: rather than repeatedly generating similar experiences, it explores an expanding region of the knowledge space. Second, the expanding coverage shows **complementary expansion**, where new experiences diversify the experience set while integrating with prior knowledge. Third, the increasing distance to initial experiences proves that co-evolution enables **genuine knowledge discovery**, as the policy learns to solve problems in ways that differ fundamentally from its SFT initialization. A frozen policy, by contrast, would be unable to generate such diverse and novel experiences, as it remains locked to its initial capability distribution.

5.2.3. EXPERIENCE USAGE PATTERNS.

Experience usage is triggered selectively, with a consistent usage rate of **4.3% to 5.0%** across training steps. This sparse usage pattern suggests the system learns to invoke experience primarily when facing genuinely challenging situations rather than relying on it uniformly. The average helpfulness score fluctuates between 0.32 and 0.41, with step 10 achieving the highest helpfulness corresponding to a period of rapid policy improvement.

5.3. Experience Impact Analysis

We analyze the causal impact of individual experiences by comparing task success rates when a given experience is retrieved versus the corresponding baseline success rates on the same tasks without experience usage. This analysis aims to assess whether retrieved experiences provide actionable guidance beyond the model's default behavior.

Among 696 experiences with sufficient usage frequency for reliable estimation, 24.1% exhibit a positive impact, de-

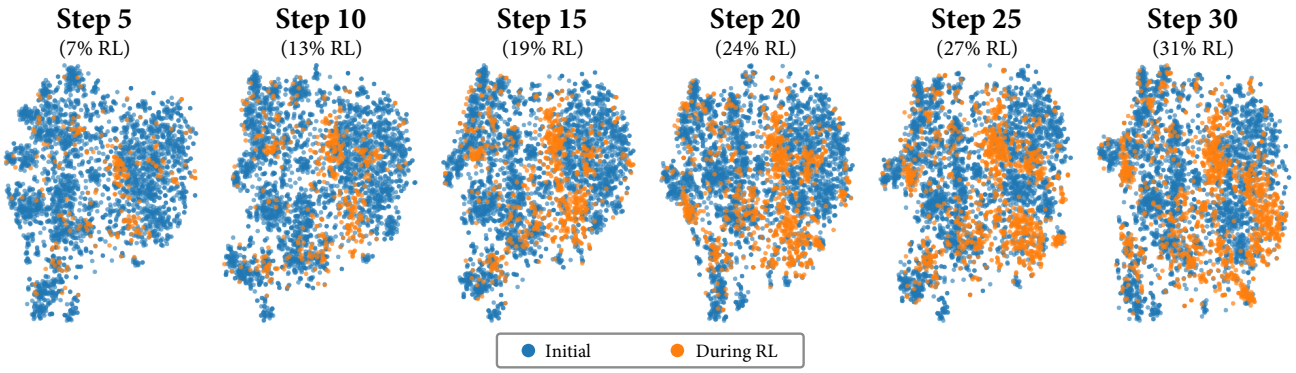

Figure 3. t-SNE visualization of experience composition over training (Steps 5 to 30). Blue points represent initialized experiences; orange points represent RL-generated experiences. The percentage of RL experiences grows from 7% to 31% as training progresses, with RL experiences integrating throughout the semantic space rather than forming isolated clusters. This demonstrates the policy's growing contribution to experience through co-evolution.

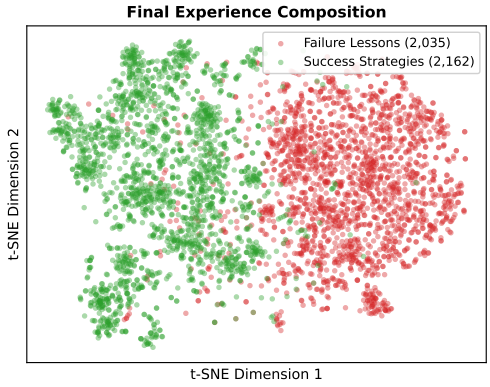

Figure 4. Experience composition by outcome type at step 31. Green: success strategies. Red: failure lessons. The two types are interleaved across the embedding space.

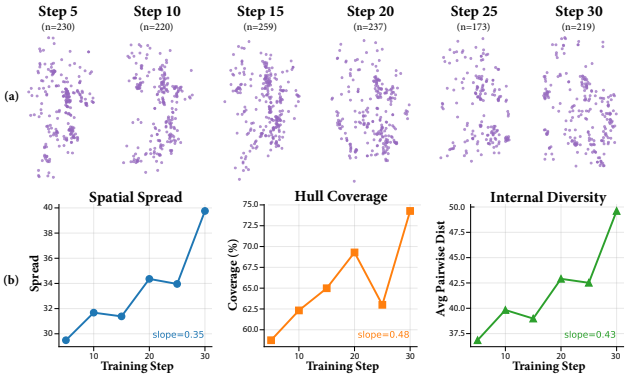

Figure 5. (a) t-SNE visualization of newly generated experiences at each training step, with consistent axis scales across panels. (b) Diversity metrics over training: spatial spread, convex hull coverage, and internal diversity all show statistically significant increasing trends, demonstrating that co-evolution enables the policy to generate progressively more diverse experiences.

fined as an increase in success rate when the experience is used, while the remaining 75.9% show a negative or neutral impact. Importantly, this asymmetry is expected due to a strong selection bias: experience retrieval is disproportionately triggered on more challenging tasks where the model anticipates difficulty. Consistent with this interpretation, the mean baseline success rate of tasks that activate experience retrieval is only 37.3%, substantially lower than the overall task success rate, indicating that experience usage is concentrated on harder instances.

Despite this skewed usage distribution, a subset of experiences demonstrates substantial positive causal effects. The highest-impact experiences achieve success-rate improvements exceeding +0.67, reaching 100% success on tasks whose baseline rates are approximately 32%. Qualitative inspection reveals that these experiences encode reusable problem-solving patterns rather than task-specific heuristics. In particular, they capture explicit verification strategies (e.g., cross-referential reasoning chains) or systematically address recurring failure modes (e.g., temporal ambiguity in historical milestones). These findings suggest that the most effective experiences function as transferable reasoning primitives, providing targeted guidance precisely in regimes where the base policy is weakest.

6. Conclusion

We presented EVOLVING ROLLOUTS, a framework that co-evolves policy parameters and an experience repository during RL training. By distilling reusable strategies from zero-variance rollout groups (all-success or all-failure), which yield no gradient signal in standard GRPO, EVOLVING ROLLOUTS recovers learning signal that would otherwise be discarded. Experiments show that EVOLVING ROLLOUTS achieves 44.7% on GAIA with a 4B model, matching the 30B reference, while ablations confirm that co-evolution is essential for optimal performance.

Impact Statement

This paper presents a reinforcement learning framework for training web-based agents with improved sample efficiency. Our work aims to advance the field of Machine Learning, particularly in the area of agentic systems that interact with real-world web environments.

We acknowledge that more capable web agents could potentially be misused for tasks such as automated misinformation gathering or unauthorized data collection. However, these concerns are not unique to our method and apply broadly to web agent research. Our framework does not introduce new capabilities beyond what existing web agents can achieve; rather, it improves the training efficiency of such systems. We believe the benefits of more sample-efficient training, which reduces computational costs and environmental impact, outweigh these general concerns.

We release our training framework to support reproducibility and encourage responsible development in this area.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report, 2023.

Cai, Z., Guo, X., Pei, Y., Feng, J., Chen, J., Zhang, Y.-Q., Ma, W.-Y., Wang, M., and Zhou, H. Flex: Continuous agent evolution via forward learning from experience. *arXiv preprint arXiv:2511.06449*, 2025.

Chae, H., Kim, S., Cho, J., Kim, S., Moon, S., Hwangbo, G., Lim, D., Kim, M., Hwang, Y., Gwak, M., et al. Web-shepherd: Advancing prms for reinforcing web agents. *arXiv preprint arXiv:2505.15277*, 2025.

Chhikara, P., Khant, D., Aryan, S., Singh, T., and Yadav, D. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Cui, G., Yuan, L., Wang, Z., Wang, H., Zhang, Y., Chen, J., Li, W., He, B., Fan, Y., Yu, T., et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Fang, R., Liang, Y., Wang, X., Wu, J., Qiao, S., Xie, P., Huang, F., Chen, H., and Zhang, N. Memp: Exploring agent procedural memory. *arXiv preprint arXiv:2508.06433*, 2025.

Geng, X., Xia, P., Zhang, Z., Wang, X., Wang, Q., Ding, R., Wang, C., Wu, J., Zhao, Y., Li, K., et al. Webwatcher: Breaking new frontier of vision-language deep research agent. *arXiv preprint arXiv:2508.05748*, 2025.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Li, K., Zhang, Z., Yin, H., Ye, R., Zhao, Y., Zhang, L., Ou, L., Zhang, D., Wu, X., Wu, J., et al. Websailor-v2: Bridging the chasm to proprietary agents via synthetic data and scalable reinforcement learning. *arXiv preprint arXiv:2509.13305*, 2025a.

Li, W., Lin, J., Jiang, Z., Cao, J., Liu, X., Zhang, J., Huang, Z., Chen, Q., Sun, W., Wang, Q., et al. Chain-of-agents: End-to-end agent foundation models via multi-agent distillation and agentic rl. *arXiv preprint arXiv:2508.13167*, 2025b.

Li, X., Jin, J., Dong, G., Qian, H., Wu, Y., Wen, J.-R., Zhu, Y., and Dou, Z. Webthinker: Empowering large reasoning models with deep research capability. *arXiv preprint arXiv:2504.21776*, 2025c.

Liang, Y., Li, J., Wang, Y., Wang, P., Tian, M., Liu, P., Qiao, S., Fang, R., Zhu, H., Zhang, G., et al. Towards personalized deep research: Benchmarks and evaluations. *arXiv preprint arXiv:2509.25106*, 2025.

Liu, J., Li, Y., Zhang, C., Li, J., Chen, A., Ji, K., Cheng, W., Wu, Z., Du, C., Xu, Q., et al. Webexplorer: Explore and evolve for training long-horizon web agents. *arXiv preprint arXiv:2509.06501*, 2025.

Mialon, G., Fourrier, C., Wolf, T., LeCun, Y., and Scialom, T. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*, 2023.

Ouyang, S., Yan, J., Hsu, I., Chen, Y., Jiang, K., Wang, Z., Han, R., Le, L. T., Daruki, S., Tang, X., et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*, 2025.

Phan, L., Gatti, A., Han, Z., Li, N., Hu, J., Zhang, H., Zhang, C. B. C., Shaaban, M., Ling, J., Shi, S., et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

Qiao, Z., Chen, G., Chen, X., Yu, D., Yin, W., Wang, X., Zhang, Z., Li, B., Yin, H., Li, K., et al. Webresearcher: Unleashing unbounded reasoning capability in long-horizon agents. *arXiv preprint arXiv:2509.13309*, 2025.

Qin, T., Chen, Q., Wang, S., Xing, H., Zhu, K., Zhu, H., Shi, D., Liu, X., Zhang, G., Liu, J., Jiang, Y. E., Gao, X., and Zhou, W. Flash-searcher: Fast and effective web agents via dag-based parallel execution, 2025. URL https://arxiv.org/abs/2509.25301.

Qwen, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Sheng, G., Zhang, C., Ye, Z., Wu, X., Zhang, W., Zhang, R., Peng, Y., Lin, H., and Wu, C. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*, 2024.

Shi, D., Cao, J., Chen, Q., Sun, W., Li, W., Lu, H., Dong, F., Qin, T., Zhu, K., Liu, M., Yang, J., Zhang, G., Liu, J., Zhang, C., Wang, J., Jiang, Y. E., and Zhou, W. Taskcraft: Automated generation of agentic tasks, 2025. URL https://arxiv.org/abs/2506.10055.

Tang, X., Qin, T., Peng, T., Zhou, Z., Shao, D., Du, T., Wei, X., Xia, P., Wu, F., Zhu, H., et al. Agent kb: Leveraging cross-domain experience for agentic problem solving. *arXiv preprint arXiv:2507.06229*, 2025.

Tao, Z., Wu, J., Yin, W., Zhang, J., Li, B., Shen, H., Li, K., Zhang, L., Wang, X., Jiang, Y., et al. Webshaper: Agentically data synthesizing via information-seeking formalization. *arXiv preprint arXiv:2507.15061*, 2025.

Wang, P., Tian, M., Li, J., Liang, Y., Wang, Y., Chen, Q., Wang, T., Lu, Z., Ma, J., Jiang, Y. E., et al. O-mem: Omni memory system for personalized, long horizon, self-evolving agents. *arXiv preprint arXiv:2511.13593*, 2025a.

Wang, Y., Takanobu, R., Liang, Z., Mao, Y., Hu, Y., McAuley, J., and Wu, X. Mem-{\alpha}: Learning memory construction via reinforcement learning. *arXiv preprint arXiv:2509.25911*, 2025b.

Wang, Z. Z., Mao, J., Fried, D., and Neubig, G. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

Wei, T., Sachdeva, N., Coleman, B., He, Z., Bei, Y., Ning, X., Ai, M., Li, Y., He, J., Chi, E. H., et al. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2025.

Wu, J., Li, B., Fang, R., Yin, W., Zhang, L., Tao, Z., Zhang, D., Xi, Z., Fu, G., Jiang, Y., et al. Webdancer: Towards autonomous information seeking agency. *arXiv preprint arXiv:2505.22648*, 2025a.

Wu, J., Li, B., Fang, R., Yin, W., Zhang, L., Wang, Z., Tao, Z., Zhang, D.-C., Xi, Z., Tang, X., Jiang, Y., Xie, P., Huang, F., and Zhou, J. Webdancer: Towards autonomous information seeking agency. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=quJdphBcdP.

Wu, J., Yin, W., Jiang, Y., Wang, Z., Xi, Z., Fang, R., Zhang, L., He, Y., Zhou, D., Xie, P., et al. Webwalker: Benchmarking llms in web traversal. *arXiv preprint arXiv:2501.07572*, 2025c.

Xbench-Team. Xbench-deepsearch, 2025. URL https://xbench.org/agi/aisearch.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report, 2025a.

Yang, C., Yang, X., Wen, L., Fu, D., Mei, J., Wu, R., Cai, P., Shen, Y., Deng, N., Shi, B., et al. Learning on the job: An experience-driven self-evolving agent for long-horizon tasks. *arXiv preprint arXiv:2510.08002*, 2025b.

Yao, Y., Zhu, H., Wang, P., Ren, J., Yang, X., Chen, Q., Li, X., Shi, D., Li, J., Wang, Q., et al. O-researcher: An open ended deep research model via multi-agent distillation and agentic rl. *arXiv preprint arXiv:2601.03743*, 2026.

Yu, H., Chen, T., Feng, J., Chen, J., Dai, W., Yu, Q., Zhang, Y.-Q., Ma, W.-Y., Liu, J., Wang, M., et al. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *arXiv preprint arXiv:2507.02259*, 2025a.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Dai, W., Fan, T., Liu, G., Liu, L., et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025b.

Zhai, Y., Tao, S., Chen, C., Zou, A., Chen, Z., Fu, Q., Mai, S., Yu, L., Deng, J., Cao, Z., et al. Agentevolver: Towards efficient self-evolving agent system. *arXiv preprint arXiv:2511.10395*, 2025.

Zhang, K., Chen, X., Liu, B., Xue, T., Liao, Z., Liu, Z., Wang, X., Ning, Y., Chen, Z., Fu, X., et al. Agent learning

via early experience. *arXiv preprint arXiv:2510.08558*, 2025.

Zhao, Y., Huang, J., Hu, J., Wang, X., Mao, Y., Zhang, D., Jiang, Z., Wu, Z., Ai, B., Wang, A., Zhou, W., and Chen, Y. Swift:a scalable lightweight infrastructure for fine-tuning, 2024. URL https://arxiv.org/abs/2408.05517.

Zheng, Y., Fu, D., Hu, X., Cai, X., Ye, L., Lu, P., and Liu, P. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *arXiv preprint arXiv:2504.03160*, 2025.

Zhou, W., Jiang, Y. E., Cui, P., Wang, T., Xiao, Z., Hou, Y., Cotterell, R., and Sachan, M. Recurrentgpt: Interactive generation of (arbitrarily) long text, 2023. URL https://arxiv.org/abs/2305.13304.

Zhu, H., Qin, T., Zhu, K., Huang, H., Guan, Y., Xia, J., Yao, Y., Li, H., Wang, N., Liu, P., Peng, T., Gui, X., Li, X., Liu, Y., Jiang, Y. E., Wang, J., Zhang, C., Tang, X., Zhang, G., Yang, J., Liu, M., Gao, X., Zhou, W., and Liu, J. Oagents: An empirical study of building effective agents, 2025. URL https://arxiv.org/abs/2506.15741.

A. Experiment Setup

A.1. Experience Extraction Details

The quality of extracted experiences is critical for effective context-space optimization. We design the extraction process to maximize transferability and minimize redundancy.

Trajectory Compression. Raw web-agent trajectories are dominated by verbose observations (e.g., full webpage HTML) and redundant intermediate reasoning, which are unsuitable for direct reuse as transferable experience. We compress each trajectory independently using a frozen LLM (Qwen3-30B-A3B (Yang et al., 2025a)), which outputs a structured JSON summary capturing the essential decision logic.

The compression schema extracts: (1) an *overall plan* summarizing the global strategy, (2) a sequence of *steps*, each containing the high-level intent, tool usage with query formulation and URL selection rationale, and a condensed observation summary, and (3) *key takeaways* encoding actionable strategy insights.

A.2. Training Data

We construct the training data for both supervised fine-tuning (SFT) and reinforcement learning (RL) by sampling questions from a diverse collection of web-agent benchmarks, including WebWalkerQA, MHQA, TaskCraft, ASearcher, WebShaper, and 2WikiMultiHopQA. From this combined pool, we randomly sample 1,000 questions for the SFT stage and a separate set of 1,000 questions for the RL stage, ensuring no overlap between the two sets. This separation prevents data leakage and ensures that the RL phase encounters novel questions that test generalization beyond the SFT distribution. The diversity of source datasets exposes the agent to a wide range of web interaction patterns, from simple fact retrieval to complex multi-hop reasoning chains.

B. Case Study: Evolving Experience

Trajectory for Experience Retrieval

Pre-retrieval State Summary:

- Goal: Extract command from last video in 2018 Replit VSCode blog post.
- Issue: Searched blog post (`https://blog.replit.com/intel`) is text-only—no video or command found.
- Hypothesis: URL may be malformed or truncated in search results.
- Blocked: Cannot verify content without correct URL.

Tool Call:

```
{
  "name": "retrieve_experience",
  "arguments": {
    "query": "How to verify the
    correct URL structure for
    Replit blog posts when search
    results show truncated
    or incorrect URLs?"
  }
}
```

Retrieved Experience (Strategy Guidance):

- **Success Strategy 1**: Direct homepage validation over secondary sources—crawl official domain to confirm content.
- **Success Strategy 2**: Standard URL pattern identification (e.g., `/post/title`) reduces ambiguity.
- **Failure Lesson 3**: Avoid broad queries without domain validation.
- **Failure Lesson 4**: Never trust URL patterns alone—always validate page content.

Evolving Experience

i). **Retrieved memory (Top-1 most relevant):**
Filter for Explicit Country Mentions Over Incidental References

ii). **New memory title:**
Historical Context Integration for Geopolitical Nuance

iii). **New memory content:**
Incorporating historical country labels (e.g., "West German") into verification prevents anachronistic misclassification.

B.1. Trajectory Compression Prompt

Trajectory Summarizer Prompt

You are an AI Trajectory Summarizer. Compress verbose web search agent logs into strategic JSON summaries.

TASK: Convert detailed trajectory to compact concise step-wise JSON capturing decision logic, eliminating redundancy.

OUTPUT JSON structure:

```
{
  "overall_view": "1-2 sentence global plan",
  "steps": [{
    "step": 1,
    "high_level_intent": "strategic goal",
    "action": {
      "tool": "tool_name_if_applicable",
      "query_strategy":
        "search formulation logic",
      "url_selection_strategy":
        "URL prioritization logic",
      "extraction_focus":
        "target information"
    },
    "observation_summary":
      "one-sentence key findings",
    "key_information_extracted":
      ["core fact"]
  }, ...],
  "key_takeaways": ["strategy insights"]
}
```

RULES:
1. Transform queries to search logic descriptions, not raw text
2. Convert observations to synthesized information states
3. Preserve goal decomposition and tool selection rationale
4. Omit system prompts, HTML, boilerplate, repetitive thinking
5. Identify successful heuristics and efficiency patterns
6. Ensure takeaways are actionable and transferable

EXAMPLE TRANSFORMATIONS:
• Query "MIT Review 2025 AI trends" →
"Combined publication, timeframe, topic for precise targeting"
• Long observation text →
"Confirmed partnership focuses on counter-drone technology"

Now compress this trajectory.

The compression rules emphasize transforming raw queries into search logic descriptions, converting verbose observations into synthesized information states, preserving goal decomposition and tool selection rationale, and omitting HTML content and repetitive reasoning. This reduces context length by 5–10× while preserving strategic content. Compression is parallelized across all trajectories in the rollout group via vLLM (Kwon et al., 2023) batched inference, minimizing wall-clock overhead.

Group Aggregation and Contrastive Prompt Design. During the RL stage, experience extraction is performed over groups of reward-labeled trajectories. These outcome labels are incorporated into contrastive prompts that explicitly contrast successful and failed attempts, or alternatively summarize common correct behaviors and recurring incorrect patterns that should be avoided.

The contrastive extraction prompt aggregates all compressed trajectories from a rollout group with their outcome labels. For failed attempts, the expected answer is included when available.

Group Extraction Prompt Template

```
You have multiple attempts to solve this problem, which may include both successful
and failed executions.  Extract 1-5 experiences that contain generalizable
strategies and lessons learned from these attempts.

PROBLEM: {question}

ATTEMPT1 (SUCCESS): {compressed_trajectory_1}
ATTEMPT2 (FAILURE): {compressed_trajectory_2}
EXPECTED: {expected_answer}
...

Extract experiences in this format:

EXPERIENCE 1:
TITLE: <concise strategy name or lesson>
DESCRIPTION: <one sentence summary>
CONTENT: <detailed transferable strategy or lesson>

Focus on:
- WHY successful approaches worked
- WHAT went wrong in failed attempts
- Generalizable patterns across attempts
```

The group extraction prompt naturally handles all outcome configurations. For all-success groups, the LLM identifies common winning strategies; for all-failure groups, it extracts anti-patterns and lessons from mistakes. This ensures every rollout group contributes to the experience repository, regardless of its utility for gradient-based policy optimization.

This ensures every rollout group contributes to the experience repository, regardless of its utility for gradient-based optimization.

Experience Embedding and Dual Representations. After extraction, each experience is embedded to support efficient semantic retrieval. Specifically, we compute two complementary dense representations per experience using a lightweight encoder (Qwen3-0.6B-Embedding (Yang et al., 2025a)): an *experience embedding* $\mathbf{v}_e \in \mathbb{R}^d$, encoded from the concatenation of the experience title and description to capture high-level strategic semantics for query-to-experience matching, and a *source problem embedding* $\mathbf{v}_e^{\text{src}} \in \mathbb{R}^d$, encoded from the original problem text to model problem-type similarity between the current task and the experience's originating context. This dual-embedding design underpins our retrieval mechanism: given an agent query representation $\mathbf{v}_q$ and a task problem representation $\mathbf{v}_p$, experiences are ranked by jointly aggregating query-to-experience similarity $\text{sim}(\mathbf{v}_q, \mathbf{v}_e)$ and problem-to-source similarity $\text{sim}(\mathbf{v}_p, \mathbf{v}_e^{\text{src}})$, ensuring that retrieved experiences are both semantically aligned with the agent's current reasoning state and contextually relevant to the target problem type.

Each extracted experience is formalized as a structured knowledge unit with a well-defined schema, designed to support reliable retrieval and effective in-context utilization. Specifically, each experience $e \in \mathscr{E}$ comprises the following fields:

Each extracted experience is formalized as a structured knowledge unit comprising both semantic content and dense vector representations for efficient retrieval.

Semantic Content. The textual component of each experience $e \in \mathscr{E}$ consists of:

- **Title**: A concise strategy identifier that abstracts the core decision pattern (e.g., "Cross-Reference Official Sources for Temporal Verification").
- **Description**: A declarative summary specifying applicability conditions and contextual triggers for the strategy.
- **Content**: A detailed reasoning template encoding transferable procedural logic and concrete guidance for in-context application.
- **Source Problem**: The original question from which the experience was extracted, preserving provenance for problem-type matching.
- **Outcome Label**: A binary indicator ($\texttt{success} \in \{0, 1\}$) denoting whether the experience derives from a successful or failed trajectory, used to distinguish winning strategies from cautionary lessons during retrieval formatting.

Retrieval Trigger. The agent autonomously decides when to retrieve based on its reasoning state. Typical triggers include:

- Encountering an unfamiliar problem type
- Reaching a reasoning impasse after failed search attempts
- Seeking verification strategies before committing to an answer

Experience Formatting. Retrieved experiences are formatted as structured guidance:

> *— Retrieved Experiences from Memory Bank —*
>
> *[Experience 1]* ***Title****: Cross-Reference Official Sources*
> ***Strategy****: When verifying institutional information, prioritize official .edu or .gov domains over third-party aggregators...*
>
> *[Experience 2] ...*
>
> *— End of Retrieved Experiences —*
>
> *Based on these strategies, refine your approach for the current task. Note: Experiences provide HOW to reason, not factual answers—verify all claims through web search.*

B.2. Training Pipeline

The complete training pipeline consists of three phases:

Phase 1: Cold-Start Repository Construction. We process N trajectories from existing web-agent datasets to construct $\mathscr{E}_0$. Each trajectory is passed through the extraction pipeline independently (without contrastive grouping, as trajectories come from different questions). This phase is executed once before training.

Phase 2: Supervised Fine-Tuning. Using the base LRM π_{θ_0} with access to $\mathscr{E}_0$, we generate SFT trajectories on a training question set. The model learns to:

- Execute the structured workflow (reason $\rightarrow$ search $\rightarrow$ scrape $\rightarrow$ answer)
- Invoke RETRIEVE at appropriate reasoning states
- Interpret and apply retrieved experiences to the current task

The SFT phase produces π_θ with basic competency in experience-augmented reasoning.

Phase 3: Reinforcement Learning with Evolution. We apply GRPO on held-out questions while dynamically evolving $\mathscr{E}$. At each step:

1. Generate m rollouts per question with retrieval from current $\mathscr{E}$
2. Compute rewards via LLM-as-a-Judge
3. **Context-space**: Extract experiences from all groups, consolidate, and forget
4. **Parameter-space**: Update π_θ via GRPO on non-zero-variance groups
5. Publish updated $\mathscr{E}$ for next iteration

Implementation Details. We implement the supervised fine-tuning (SFT) stage using the `ms-swift` (Zhao et al., 2024) framework, which provides scalable support for instruction tuning and experience-augmented data pipelines on large reasoning models. The reinforcement learning stage is implemented with `veRL` (Sheng et al., 2024), a distributed RL framework that supports group-based policy optimization (GRPO) with efficient rollout generation and reward aggregation. All experiments are conducted on NVIDIA A800 GPUs, with a total compute budget of approximately 800 GPU-hours.

B.3. Computational Considerations

Extraction Overhead. Experience extraction adds computational cost, but this is amortized across future rollouts that benefit from the enriched repository. We extract experiences asynchronously with policy updates to minimize wall-clock overhead.

Repository Size Management. The merge threshold θ controls repository growth. With $\theta = 0.9$, we observe approximately linear growth early in training (diverse experiences) followed by sublinear growth as the repository saturates with high-quality strategies and new experiences increasingly merge with existing ones.

B.4. Hyperparameters

Table 3 summarizes key hyperparameters for the evolution mechanisms.

Table 3. Hyperparameters for experience evolution.

Parameter	Symbol	Value
Merge threshold	θ	0.90
Retrieval top-k	k	5
Dual similarity weight	α	0.5
Success threshold	-	0.5
Experiences per extraction	-	1–5
Rollouts per question (GRPO)	m	8

Sensitivity Analysis. The merge threshold θ presents a trade-off: higher values preserve more distinct experiences but risk redundancy; lower values aggressively consolidate but may lose nuance. We find $\theta \in [0.85, 0.95]$ works well across benchmarks. The forgetting mechanism is intentionally conservative (requiring unanimous failure) to prevent premature deletion of potentially useful strategies.

C. Software Frameworks and Licenses

Table 4. Software components and licenses.

Component	Role	License
`ms-swift` (Zhao et al., 2024)	SFT	Apache License 2.0
`veRL` (Sheng et al., 2024)	RL	Apache License 2.0
`Qwen-3` (Yang et al., 2025a)	Backbone	Apache License 2.0

This section summarizes the major frameworks used in our training pipeline and their corresponding licenses. All components are selected to ensure reproducibility and compliance with open-source research practices.