SlimSearcher: Training Efficiency-Aware Web Agents via Adaptive Reward Gating

Anonymous ACL submission

Abstract

Deep research agents have demonstrated remarkable capabilities in complex information-seeking tasks, yet this power comes at a steep computational cost. Driven by accuracy-focused training paradigms, current models adopt brute-force strategies characterized by *blind tool dependency* and *performative reasoning*—generating long, redundant trajectories that are far from necessary for resolving these tasks, leading to wasteful tool calls and excessive token consumption. To overcome this efficiency trap, we propose **SlimSearcher**, a principled framework that pushes the Pareto frontier between accuracy and computational cost across both Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL). In the SFT stage, SlimSearcher employs Pareto-efficient filtration to distill trajectories that are both successful and economical, guiding the model toward inherently efficiency-aware search behaviors. During RL, we introduce *Adaptive Reward Gating*, a dynamic reward-shaping mechanism that evaluates relative tool and token efficiency within a sampled cohort. By cascading these adaptive efficiency metrics with a strict correctness gate, our approach effectively avoids the brevity bias associated with absolute penalties and mitigates reward hacking. Extensive experiments on long-horizon benchmarks, including GAIA, BrowseComp, and XBench-DeepSearch, demonstrate that SlimSearcher reduces average tool-call rounds by 17%–58% while maintaining or improving accuracy.

1 Introduction

With the rapid advancement of artificial intelligence, LLMs have progressed from static text generators to sophisticated agents that can leverage external tools and interact with dynamic environments (Bai et al., 2025b; Zeng et al., 2025). By integrating language understanding with external tools such as search engines and code interpreters, tool-integrated reasoning substantially expands the problem-solving capabilities of LLMs beyond purely linguistic reasoning. This capability not only improves adaptability in open-ended tasks but also supports a new generation of powerful systems, ranging from commercial offerings such as the Deep Research feature of OpenAI (OpenAI, 2025a) to emerging open-source agents such as Tongyi-DeepResearch (Li et al., 2025a) and MiroThinker (Bai et al., 2025a).

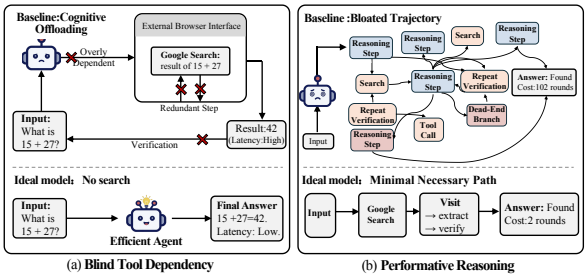

Figure 1: **Behavioral Analysis of the Efficiency Trap.** (a) **Blind Tool Dependency**: The baseline agent indiscriminately invokes external search tools for a common-sense query resolvable via internal knowledge, leading to increased latency. (b) **Performative Reasoning**: As demonstrated in the complex query case (see Figure 8), even in a successful trajectory, the baseline agent generates redundant loops and dead-end branches.

However, existing open-source web agents remain highly inefficient. For example, systems such as MiroThinker often require several hundred reasoning rounds to answer a single question. This inefficiency arises from training paradigms that incentivize brute-force strategies, in which task success is pursued by indiscriminately increasing tool usage. The resulting volume of unnecessary tool calls imposes substantial computational and time overhead: executing tools at scale requires considerable infrastructure resources and increases operational costs.

Our analysis attributes this overall inefficiency to two distinct failure modes that arise at different

stages of the agent workflow: *blind tool dependency* and *performative reasoning*. First, **blind tool dependency** reflects a failure of initiation and functions as a form of indiscriminate cognitive offloading (Risko and Gilbert, 2016). Agents not only overuse tools but also invoke external tools for simple queries that can be resolved utilizing internal knowledge, thereby consuming API calls redundantly and introducing unnecessary latency rather than directly producing an answer. Second, **performative reasoning** emerges in complex tasks that legitimately require external tools and represents a critical failure of execution efficiency. Rather than identifying a direct solution, the reasoning process of the agent becomes saturated with redundant loops, repetitive verifications of previously established facts, and unproductive exploratory branches. Consequently, models generate protracted trajectories that mimic rigor yet add no substantive information to the final answer. Collectively, these issues indicate a fundamental deficiency in the ability of the model to extract the Minimal Necessary Path (Wang et al., 2026).

We find that these defects are exacerbated by the prevailing **Acc-only Rejection Sampling** paradigm, where any correct trajectory, regardless of its redundancy, is treated as a positive sample for SFT. Naively training on such data causes models to internalize bloated patterns (see our experiments), as standard rejection sampling provides no mechanism to filter out these hollow trajectories. Furthermore, existing Reinforcement Learning (RL) strategies for web agents(Jin et al., 2025; Li et al., 2025e) primarily incentivize end-to-end success rates. This singular focus triggers an "efficiency collapse" during RL training: as iterations progress, the model tends to scale up search rounds and context length as a brute-force hedge to maximize accuracy.

To address this efficiency trap, we propose **SlimSearcher**, a framework designed for long-horizon deep research scenarios that systematically integrates efficiency optimization into both SFT and RL to push the Pareto frontier between accuracy and computational cost. Our core philosophy is that the system must not only guide the model to the correct answer but also progressively converge toward the Minimal Necessary Path. Specifically, our contributions are summarized as follows:

- **Efficiency-Aware SFT:** Instead of filtering trajectories by correctness alone, we introduce a joint efficiency evaluation to construct a refined training dataset. This ensures the model learns efficient search behaviors rather than redundant patterns.
- **Adaptive Efficiency Anchoring (AEA) for Reward Shaping:** To circumvent the brevity bias inherent in fixed penalties, we design a dynamic reward structure that explicitly anchors optimization to the empirical Minimal Necessary Path discovered within a sampled trajectory group. This approach provides a task-adaptive gradient signal, incentivizing convergence toward minimal trajectory complexity without compromising task accuracy.
- **Efficiency-Accuracy Pareto Improvement:** Extensive experiments on challenging long-horizon benchmarks, including GAIA, BrowseComp, and XBench-DeepSearch, demonstrate SlimSearcher's strong efficiency gains. Compared to baselines, SlimSearcher reduces average tool-call rounds by **17%–58%** while improving task accuracy.

2 Related Work

2.1 Paradigm of Web Agents

Current methodologies for autonomous web agents can be broadly categorized into two streams. The first stream is *Prompting and Engineering Frameworks*, which elicit agentic behaviors from frozen LLMs through collaborative architectures. This includes classical prompting strategies like SelfDC (Wang et al., 2025), as well as multi-agent frameworks such as AutoGen and GPT-Researcher. While flexible, these systems rely heavily on the inherent capabilities of the base model and often incur high token costs due to verbose context management. The second stream is *Training-Centric Agent Evolution*, which internalizes search and reasoning capabilities directly into model parameters via Supervised Fine-Tuning (SFT) (Tao et al., 2025) or Reinforcement Learning (RL). Early works focused on synthesizing trajectories for SFT, such as WebSailor (Li et al., 2025b) and WebShaper (Tao et al.). More recent approaches leverage RL to enhance performance, as seen in Search-R1 (Jin et al., 2025). However, most training-based methods prioritize success rates as the sole metric, often incentivizing agents to adopt "brute-force" strategies that exhaust computational resources to force

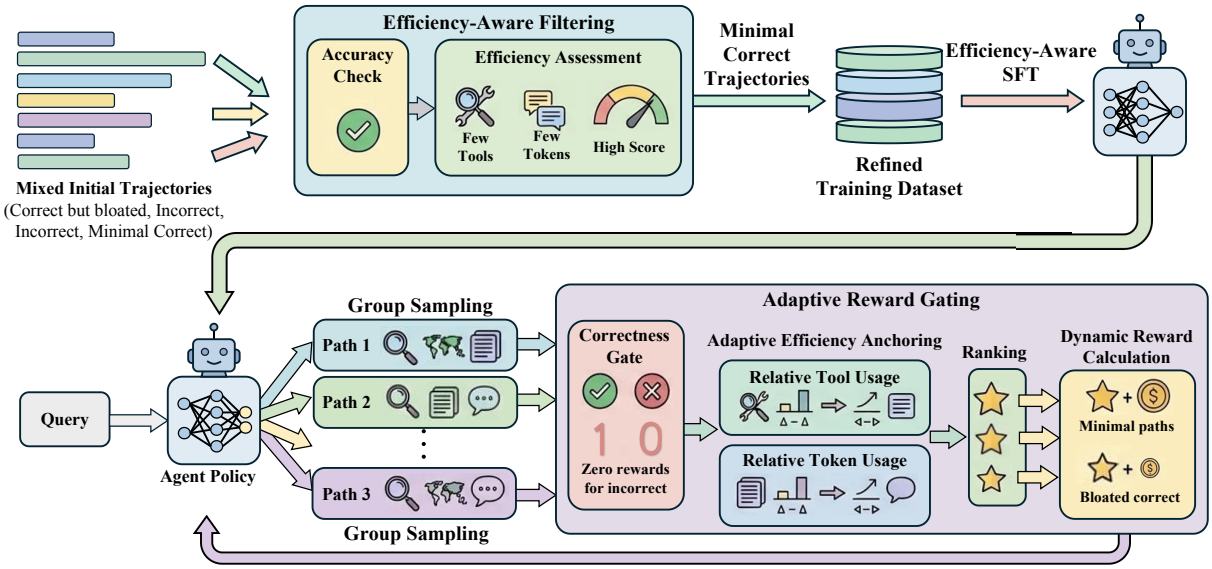

Figure 2: SlimSearcher: Multi-dimensional SFT filtering and adaptive reward gating enable web agents to learn efficiency-aware and correct search behaviors.

a correct answer, largely ignoring inference efficiency and computational cost. Existing methods reduce internal reasoning length through prompt engineering (Han et al., 2025; Xu et al., 2025) or by penalizing long Chain-of-Thought (CoT) paths during RL (Ma et al., 2025; Munkhbat et al., 2025; Aggarwal and Welleck, 2025). However, optimizing web agents requires more than reducing text tokens; it requires systematically pruning redundant external actions—such as cyclic searches or unproductive browsing—which are the primary bottleneck in web agents.

2.2 Efficiency in Tool-Integrated Reasoning

As agents tackle increasingly complex, long-horizon tasks, execution efficiency has emerged as a critical dimension. In the realm of SFT, **WebLeaper** (Tao et al., 2025) argues that sparse entity tasks in training data limit the agent's ability to learn efficient search behaviors, proposing the synthesis of high-density information-seeking tasks to address this gap. Unlike WebLeaper, which focuses primarily on data synthesis, **SlimSearcher** fundamentally advances this paradigm by unifying efficiency optimization across the entire training pipeline . To achieve this, SlimSearcher introduces **Adaptive Efficiency Anchoring (AEA)**. Instead of applying static cost constraints that often induce brevity bias, AEA dynamically calibrates the reward landscape by anchoring it to the empirical minimal necessary path discovered within a sampled trajectory group. Furthermore, we enforce a strict correctness gate to inherently prevent reward hacking. This synergistic design ensures that SlimSearcher autonomously shifts the Pareto frontier, rigorously preserving task accuracy while systematically eradicating computational pleonasm.

3 Methodology

In this section, we present the **SlimSearcher** framework. Unlike previous methods that rely on heuristic prompt engineering, SlimSearcher unifies the optimization of accuracy and efficiency into a single **Multi-Stage Gating** mechanism. This mechanism serves a dual purpose: it acts as a *Pareto Filter* during the SFT phase to construct high-signal demonstration data via Rejection Sampling, and as a *Cascading Reward Function* during the RL phase to guide policy evolution.

3.1 The Multi-Stage Gating Mechanism

The core of SlimSearcher is a hierarchical valuation system. Its primary objective is to purge "unproductive trajectories"—paths that are technically correct but marred by *blind tool dependency* or *performative reasoning*. Drawing inspiration from the convergence mechanisms in ant colony optimization, our reward shaping is designed to iteratively approximate the Minimal Necessary Path . By progressively driving the policy toward the shortest viable route over successive explorations, we ensure agents operate on the efficient Pareto frontier. Formally, given a question q and a generated trajectory τ, the final reward R_{final} is computed via

a multiplicative cascading logic:

$$R_{final}(\tau) = r_{correct}(\tau) \cdot r_{tool}(\tau) \cdot r_{len}(\tau) \quad (1)$$

3.1.1 Gate 1: The Correctness Gate

The first gate imposes a strict binary constraint. We define $r_{correct}$ as:

$$r_{correct} = \mathbb{I}(\text{Answer}(\tau) \approx y_{gt}) \quad (2)$$

If the trajectory fails to recover the ground truth y_{gt}, $r_{correct} = 0$, forcing the entire chain R_{final} to zero. This ensures that the model never prioritizes brevity over accuracy, strictly preventing the "reward hacking" phenomenon where models produce short but hallucinatory answers.

3.1.2 Gate 2: Adaptive Anchoring for Tool Efficiency

For trajectories that pass the Correctness Gate, we introduce an adaptive scaling mechanism for tool usage. Instead of applying a fixed linear penalty to tool calls—which often provides weak gradient signals (Wu, 2024) and induces *brevity bias* (Zhang et al., 2024)—we dynamically anchor the reward to the most efficient trajectory discovered within the current exploration batch. Analogous to ant colony optimization, where the shortest discovered route receives the strongest reinforcement, we use the empirical minimum cost to shape the reward landscape.

Specifically, a trajectory τ_i consists of a sequence of reasoning steps and actions. We denote the set of external tool invocations within τ_i as $\mathcal{A}(\tau_i)$. The total tool cost $C(\tau_i)$ is formulated as the weighted sum of all executed tool calls:

$$C(\tau_i) = \sum_{a \in \mathcal{A}(\tau_i)} w_{type}(a) \quad (3)$$

where $w_{type}(a)$ represents the predefined penalty weight corresponding to the specific tool type of action a. For simplicity, we set $w_{type}(a) = 1$ uniformly for all tool types in our experiments.

Given a candidate set of sampled trajectories for the same query, we identify the empirical minimum cost among the successful paths as C_{min}. To quantify inefficiency relative to this "shortest path," we define the relative deviation δ_i and the corresponding score S_i:

$$\delta_i = \frac{C(\tau_i) - C_{min}}{C_{min} + \epsilon}, \quad S_i = -\delta_i \quad (4)$$

where ϵ is a small constant for numerical stability (e.g., 10^{-8}). Rather than using S_i directly, we map this relative score into a bounded reward space using an exponential transformation:

$$r_{tool}(\tau_i) = 2 \cdot \frac{\exp(S_i)}{\exp(S_i) + \exp(S_{opt})} \quad (5)$$

where $S_{opt} = 0$ represents the optimal behavior in the candidate set (i.e., when $C(\tau_i) = C_{min}$).

This design choice is deliberate: much like pheromone trails that dissipate rapidly over suboptimal routes, the non-linear mapping ensures that trajectories diverging from the empirical optimal path suffer an exponentially decaying penalty. By converting absolute tool costs into a strictly bounded multiplier ($r_{tool} \in (0, 1]$), we provide a sharper and more stable efficiency signal prior to the advantage estimation computed later during the RL phase.

3.1.3 Gate 3: Adaptive Anchoring for Token Efficiency

While Gate 2 optimizes for external tool costs, it does not prevent the model from generating excessive internal tokens to compensate for fewer observations. To address this *performative reasoning*, we introduce the **Adaptive Efficiency Anchoring** mechanism to impose a dynamic length constraint relative to the candidate set's most concise successful solution.

Let $\mathcal{G}_{correct}$ denote the subset of concurrent trajectories that passed Gate 1. We define the baseline minimal length L_{min}:

$$L_{min} = \min_{\tau \in \mathcal{G}_{correct}} L(\tau) \quad (6)$$

For any trajectory τ_i, we calculate the relative length deviation ρ_i and the score $S_{len,i}$, quantifying the redundant token consumption:

$$\rho_i = \frac{L(\tau_i) - L_{min}}{L_{min} + \epsilon}, \quad S_{len,i} = -\rho_i \quad (7)$$

Applying the same non-linear transformation used in Gate 2, we map this score into a bounded multiplier:

$$r_{len}(\tau_i) = 2 \cdot \frac{\exp(S_{len,i})}{\exp(S_{len,i}) + \exp(S_{opt})} \quad (8)$$

where $S_{opt} = 0$. This parallel formulation guarantees that overly verbose trajectories face a strict, non-linear penalization proportional to their excess length. By continuously reinforcing the most

compact generation within the competitive cohort, the policy progressively suppresses performative bloated patterns, driving the agent toward truly parsimonious reasoning.

3.2 Efficiency-Aware SFT via Reward-Guided Rejection Sampling

We employ the base model as our backbone. While this model possesses robust reasoning capabilities, its raw outputs often suffer from *verbosity bias*—generating trajectories that are factually correct yet computationally inefficient. To align the model with our efficiency objectives, we construct a high-signal dataset, $\mathcal{D}_{sft}$, through a rigorous *Pareto-Efficient Filtration* pipeline, acting as a reward-guided Rejection Fine-Tuning process.

Specifically, we compile a comprehensive seed corpus comprising 13,863 high-quality trajectories sourced from a diverse array of information-seeking datasets (A.3), alongside synthetically generated instances. To ensure the SFT phase targets non-trivial reasoning, we evaluate query difficulty by executing each prompt four times in the base environment. We retain only queries q exhibiting a pass rate $\mathrm{PR}(q)$ within the interval $0 < \mathrm{PR}(q) < 1$, effectively excluding trivial queries requiring no deep research and impossible tasks that might introduce noise. For each filtered query q, we sample a candidate set of K trajectories, denoted as $\mathcal{T}_q = \{\tau_1, \tau_2, \ldots, \tau_K\}$, using the base model as the generator, and apply our Multi-Stage Gating mechanism. First, we filter out all trajectories that fail to recover the ground truth by applying the correctness indicator, yielding a valid subset $\mathcal{T}_q^{valid}$:

$$\mathcal{T}_q^{valid} = \{\tau \in \mathcal{T}_q \mid r_{correct}(\tau) = 1\} \tag{9}$$

Subsequently, we evaluate the remaining valid candidates based on their joint efficiency score, defined as the product of the tool-call efficiency r_{tool} and the token efficiency r_{len}. We distill the Minimal Necessary Path (MNP), denoted as τ^*, by selecting the trajectory that maximizes this joint objective:

$$\tau^* = \arg\max_{\tau \in \mathcal{T}_q^{valid}} \big(r_{tool}(\tau) \times r_{len}(\tau)\big) \tag{10}$$

This rigorous filtration effectively purges redundant or poisoned reasoning loops. The final collection of these optimal trajectories τ^* forms our refined dataset $\mathcal{D}_{sft}$, which is explicitly optimized to teach parsimonious and cost-aware search behaviors right from the initialization stage.

3.3 Policy Optimization with Multi-Stage Gating Reward

We further employ GRPO(Guo et al., 2025)-based reinforcement learning with the Multi-Stage Gating reward to guide the model in exploring the Pareto-optimal frontier between efficiency and accuracy.

For each input question $q \sim \mathcal{D}$, we sample a group of G trajectories $\{\tau_1, \tau_2, \ldots, \tau_G\}$ from the old policy π_{old}. The raw reward for each trajectory is computed using our cascading mechanism: $R_i = R_{final}(\tau_i)$, which is the same as Equation (1).

We then estimate the group-relative advantage $\hat{A}_i$ for each trajectory by standardizing its reward against the group's performance:

$$\hat{A}_i = \frac{R_i - \mathrm{mean}(\{R_1, \ldots, R_G\})}{\mathrm{std}(\{R_1, \ldots, R_G\}) + \epsilon_s} \tag{11}$$

where ϵ_s is a small constant for numerical stability. This trajectory-level advantage $\hat{A}_i$ is applied to every generated token at timestep t within the trajectory τ_i. The policy model π_θ is updated by maximizing the clipped surrogate objective function:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}_{q\sim\mathcal{D},\{\tau_i\}_{i=1}^{G}\sim\pi_{old}} \Bigg[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|\tau_i|} \sum_{t=1}^{|\tau_i|} \min\Big(p_{i,t}\hat{A}_i, \mathrm{clip}(p_{i,t}, 1-\epsilon_c, 1+\epsilon_c)\hat{A}_i\Big) \Bigg] \tag{12}$$

where $p_{i,t} = \frac{\pi_\theta(y_{i,t}|q, y_{i,<t})}{\pi_{old}(y_{i,t}|q, y_{i,<t})}$ represents the importance sampling ratio of generating token $y_{i,t}$ given the query q and preceding tokens $y_{i,<t}$ at timestep t, ϵ_c is the clipping bound to ensure stable updates.

4 Experiments

4.1 Experimental Settings

Datasets. We conduct experiments on four widely adopted web agent benchmarks: XBench-DeepSearch (Xbench Team, 2025), BrowseComp (Wei et al., 2025), GAIA (Mialon et al., 2023), and HLE (Phan et al., 2025), following the same set as (Wang et al., 2026). More details can be found in Appendix A.5

Evaluation Metrics. To comprehensively assess the trade-off between task success and operational expenditure, we employ a multi-dimensional framework. We measure **Accuracy (Acc)** as the primary success rate, representing the percentage of queries where the final response aligns with the ground truth. To quantify efficiency, we track **Tool-Call Rounds** , the average number of external tool

Method	XBench-DeepSearch			Browsecomp			GAIA			HLE		
	Acc ↑	Rounds ↓	Token ↓	Acc ↑	Rounds ↓	Token ↓	Acc ↑	Rounds ↓	Token ↓	Acc ↑	Rounds ↓	Token ↓
Proprietary Systems												
OpenAI o3	0.670	-	-	0.497	-	-	-	-	-	0.249	-	-
OpenAI DeepResearch	-	-	-	**0.515**	-	-	0.674	-	-	0.266	-	-
Claude-4-Sonnet	0.646	-	-	0.122	-	-	0.683	-	-	0.203	-	-
Open-weights Agents												
Kimi-K2-Instruct-0905	0.540	5.98	1316	0.094	16.65	3426	0.469	6.45	1281	0.146	**5.17**	**2349**
Qwen3-235B-A22B-Instruct	0.490	8.84	938	0.046	13.70	1837	0.456	7.14	1128	0.199	7.45	2960
DeepSeek-V3	0.530	**2.23**	**422**	0.038	**4.78**	**1317**	0.373	**3.20**	994	0.060	13.00	2713
WebExplorer	0.517	9.05	2246	0.137	29.43	6289	0.372	12.88	3560	0.116	15.52	6579
WebLeaper	0.780	14.06	4466.50	0.388	60.07	10894	0.702	16.65	5045	0.298	24.86	11924
Qwen3-30B-A3B-Instruct-2507*	0.550	15.04	2395	0.029	25.80	1683	0.339	11.63	**986**	0.009	25.77	2570
Baseline: Qwen3-30B-A3B-Instruct-2507												
SlimSearcher (SFT)	0.740	13.44	5515	0.309	65.53	12185	0.665	24.46	7299	0.259	27.86	13169
SlimSearcher (SFT+RL)	0.770	10.87	5092	0.328	47.22	10883	0.699	20.13	7237	0.278	19.51	11497
Tongyi-DeepResearch	0.713	14.26	6918	0.410	63.70	12014	0.682	20.56	7378	0.358	23.92	13664
Baseline: Tongyi-DeepResearch												
Prompt Control	0.676	12.50	6321	0.373	62.80	12222	0.663	18.70	6752	0.349	23.91	14107
SlimSearcher (SFT)	0.770	10.22	5155	0.428	61.28	11522	0.660	21.79	6412	0.366	27.16	12936
SlimSearcher (SFT+RL)	**0.790**	5.92	4888	0.447	47.63	11093	**0.709**	10.61	4915	**0.376**	19.82	11418

Table 1: **Main results across diverse web agent benchmarks.** Evaluation metrics include task accuracy (Acc ↑), interaction rounds (Rounds ↓), and total token consumption (Token ↓). **Bold** and underline indicate the best and second-best results, respectively, across all evaluated models for each metric. Our proposed method (**SlimSearcher**) demonstrates a Pareto improvement in both task completion and resource economy across all benchmarks. * denotes results evaluated in our unified environment.

invocations per query, which directly reflects the model's ability to mitigate *blind tool dependency*. Additionally, we monitor **Token Efficiency** via the average consumption of reasoning tokens per query. This metric serves as a key indicator of *performative reasoning* and overall computational overhead.

Baselines. Our comparison includes both closed-source and open-source agents. Closed-source systems include OpenAI o3 (OpenAI, 2025b), OpenAI DeepResearch (OpenAI, 2025a), and Claude-4-Sonnet (anthropic, 2025), with test results cited from their official reports. The open-source agents include Kimi-K2-Instruct-0905 (Bai et al., 2025b), Qwen3-235B-A22B-Instruct-2507 (Yang et al., 2025), DeepSeek-V3 (DeepSeek-AI et al., 2025), WebExplorer (Liu et al., 2025), and WebLeaper (Tao et al., 2025). To verify the effectiveness of our approach, we design the following baseline: *Prompt Control*, where we add instructions to the agent's system prompt, explicitly asking it to avoid irrelevant information and repetitive validation while controlling the number of tool calls.

Implementation. Our backbone policies build on *Tongyi-DeepResearch-30B* and *Qwen3-30B-A3B-Instruct-2507*. We distribute computation across 64 NVIDIA H800 GPUs for SFT and 64 H800 GPUs for RL to meet the high rollout throughput required by GRPO. For environment interaction, we use the Serper API for search and Jina Reader for URL parsing. To improve statistical reliability, we run each evaluation three times and report the average Pass@1 and efficiency metrics. Detailed hyperparameters are provided in Appendix A.

4.2 Main Results

Table 1 reports the performance of SlimSearcher and baseline models on benchmarks. The results show that SlimSearcher improves efficiency while maintaining accuracy across different backbones.

For the **Tongyi-DeepResearch** backbone, SlimSearcher (SFT+RL) reduces computational overhead while maintaining or improving accuracy. On GAIA, tool-call rounds decrease by 48.4% (from 20.56 to 10.61) and token usage decreases by 33.4%, while accuracy increases from 0.682 to 0.709. On Browsecomp, tool-call rounds decrease from 63.70 to 47.63 and token usage decreases from 12014 to 11093, while accuracy increases from 0.410 to 0.447.

For the **Qwen3-30B-A3B-Instruct** backbone, the base model shows low accuracy because the search is not sufficiently deep. Fine-tuning on our

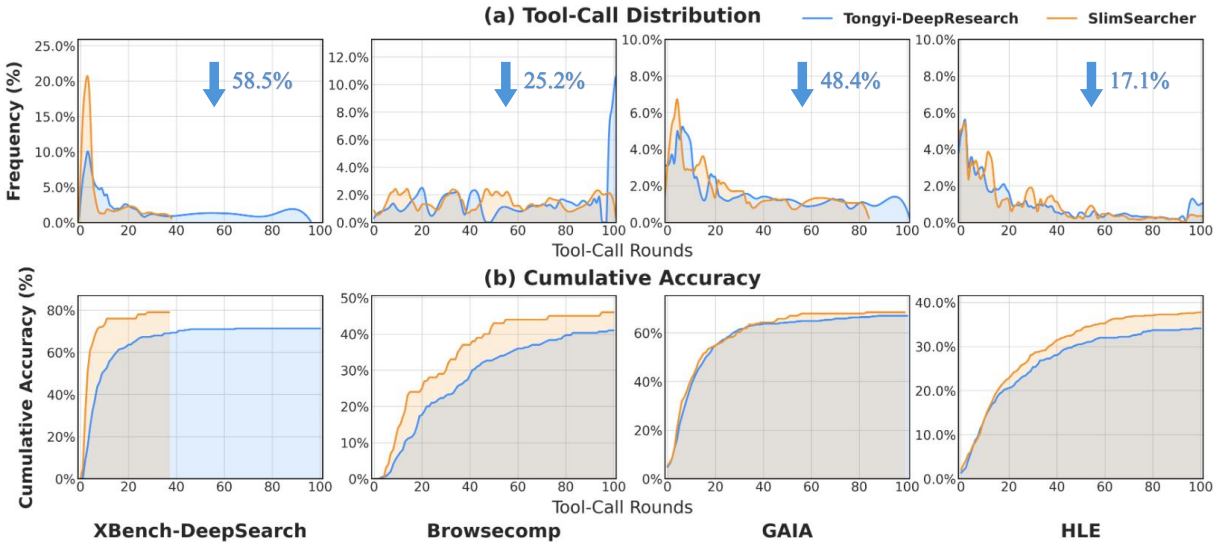

Figure 3: Comparison of tool-call distribution and cumulative accuracy.

high-quality, Pareto-filtered dataset enables deeper research behavior and improves accuracy . This stage increases token usage to 7299 because the model learns to execute more complex searches, but the subsequent RL stage shortens the trajectory. On HLE, the RL stage reduces tool-call rounds from 27.86 to 19.51 and increases accuracy from 0.259 to 0.278.

In contrast, Prompt Control fails to consistently improve efficiency and actually reduces accuracy. Similarly, baselines like DeepSeek-V3 consume minimal tokens but fail to achieve competitive accuracy. Overall, these results demonstrate that coupling Pareto-filtered SFT data with Adaptive Reward Gating enables SlimSearcher to prune redundant tool calls without sacrificing task accuracy.

4.3 Detailed Performance Analysis Across Benchmarks

To further assess the robustness and efficiency of SlimSearcher, we conduct a fine-grained analysis of the distribution of search rounds and cumulative accuracy across four representative benchmarks. As shown in Figure 3, SlimSearcher consistently shifts the Pareto frontier, achieving higher accuracy with a substantially more compact action space.

GAIA: Strict Execution Discipline in Complex Reasoning The GAIA dataset requires complex, multi-step reasoning. Notably, in such reasoning-intensive scenarios, many queries can actually be resolved utilizing the model's internal knowledge without relying on external tools. However, baseline models struggle significantly here, exhibiting "blind tool dependency" and scattered, inefficient tool usage when internal capabilities would suffice. In contrast, SlimSearcher enforces strict reasoning discipline. By mitigating this indiscriminate cognitive offloading , it resolves most queries on GAIA within the 0–20 round bucket, driving a steep accuracy curve that approaches 70%. This demonstrates the framework's ability to effectively leverage its inherent capabilities, eliminate invalid trial-and-error steps, and strategically target core information through external tools only when necessary.

XBench, Browsecomp, and HLE: Eliminating the Efficiency Trap in Exploration Beyond deep reasoning, web agents also face severe challenges in open-domain search, multi-step page navigation, and extreme long-horizon tasks, which SlimSearcher consistently overcomes across the remaining benchmarks. The XBench dataset involves open-ended web search and information extraction. On this dataset, the baseline model's tool-call distribution exhibits a long, inefficient tail, indicating that the agent frequently gets caught in unproductive verification loops and cyclic behavior. SlimSearcher markedly compresses this distribution, sharply concentrating almost all tool calls within the 0–20 round interval with a massive peak under 10 rounds. The cumulative accuracy curve shows that SlimSearcher reaches peak performance early, which confirms that the proposed framework removes performative reasoning loops while improving task success. Similarly, Browsecomp requires multi-step web navigation, where agents can easily get lost in irrelevant exploratory branches.

Method	XBench-DeepResearch			Browsecomp			GAIA			HLE		
	Acc ↑	Rounds ↓	Token ↓	Acc ↑	Rounds ↓	Token ↓	Acc ↑	Rounds ↓	Token ↓	Acc ↑	Rounds ↓	Token ↓
Base Model												
Qwen3-30B-A3B-Instruct	0.550	15.04	2395	0.029	25.80	1683	0.339	11.63	986	0.009	25.77	2570
Stage 1: SFT Phase												
SFT (Standard Rejection Sampling)	0.730	14.08	5338	0.287	66.08	12065	0.641	25.90	7478	0.262	29.35	12724
SFT (Reward-Guided Rejection Sampling)	0.740	13.44	5515	0.309	65.53	12185	0.665	24.46	7299	0.259	27.86	13169
Stage 2: RL Phase												
w/o Correctness Gate	0.440	1.42	3022	0.007	0.20	7395	0.136	0.07	4239	0.124	1.17	4898
w/o Adaptive Efficiency Anchoring	0.780	15.54	5671	0.339	74.10	13644	0.641	25.42	7409	0.283	31.05	13781
SlimSearcher (Full)	0.770	10.87	5092	0.328	47.22	10883	0.699	20.13	7237	0.278	19.51	11497

Table 2: Performance comparison of different training strategies across four long-horizon web research benchmarks.

The distribution for both the Base models shows a long, inefficient tail, with a massive spike of failures hitting the > 100 round budget. SlimSearcher suppresses this long tail. It densely concentrates its tool calls in the 20–60 round interval, and its cumulative accuracy at 40 rounds already eclipses the final accuracy of the baseline at 100 rounds. Finally, HLE requires complex, multi-step reasoning where baseline models suffer complete failure by hitting the > 100 round limit with flat accuracy. On the inherently difficult HLE dataset, SlimSearcher bounds its exploration within the first 40 rounds, maintaining a steady accuracy growth that consistently dominates the baselines.

4.4 Ablation Study

Effectiveness of Reward-Guided Rejection Sampling. Unlike *Standard Rejection Sampling*, which indiscriminately accepts any correct trajectory and inadvertently internalizes bloated reasoning patterns, our *Reward-Guided* approach enforces a rigorous joint efficiency evaluation. As demonstrated in Table 2, this mechanism yields consistent Pareto improvements across benchmarks. For instance, on GAIA, it not only elevates task accuracy ($0.641 \rightarrow 0.665$) but also systematically compresses the trajectory footprint, reducing average tool-call rounds ($25.90 \rightarrow 24.46$) and token consumption ($7478 \rightarrow 7299$).

Necessity of the Correctness Gate. Removing the strict correctness constraint during RL induces a catastrophic accuracy collapse accompanied by an anomalous plunge in tool usage. As shown in Table 2, without this gate, the agent's performance on GAIA drops significantly in accuracy (0.136) while registering near-zero tool exploration (0.07 rounds). This indicates severe reward hacking: in the absence of an absolute success prerequisite, the policy optimizes solely for the efficiency reward by generating immediate, vacuous responses, bypassing the reasoning process entirely.

Impact of Adaptive Efficiency Anchoring. The ablation of our dynamic scaling mechanism (*w/o Adaptive Efficiency Anchoring*) severely disrupts the delicate equilibrium between task efficacy and computational cost. Without the empirical minimal path serving as a dynamic anchor, the model defaults to brute-force exploration. On HLE, this unrestricted search space inflates tool-call rounds significantly ($19.51 \rightarrow 31.05$) to squeeze out a marginal accuracy gain , incurring computational overhead. Furthermore, the absence of this joint anchoring degrades internal reasoning discipline; on reasoning-intensive benchmarks like GAIA, accuracy actually deteriorates ($0.699 \rightarrow 0.641$) despite increased tool usage ($20.13 \rightarrow 25.42$), underscoring the necessity of AEA in suppressing performative reasoning.

5 Conclusion

In this paper, we address the efficiency trap in web agents by proposing **SlimSearcher**, a training framework driven by a **Multi-Stage Gating** mechanism. Unlike traditional methods that treat efficiency as a secondary constraint, our approach unifies supervised fine-tuning and reinforcement learning under a single objective. Specifically, it employs a strict correctness gate and uses **Adaptive Reward Gating** to dynamically optimize computational costs. Extensive experiments on long-horizon benchmarks demonstrate that **SlimSearcher** reduces average tool-call rounds by 17% to 58% without compromising task accuracy. This transforms open-source models into accurate and computationally disciplined agents, offering a scalable approach for developing robust and highly efficient next-generation web agents.

Limitations

Despite its superior efficiency and accuracy, **SlimSearcher** has several limitations that warrant future research. First, our current framework is optimized for text-based reasoning; as web environments become increasingly visual, extending the **Adaptive Efficiency Anchoring (AEA)** mechanism to handle multimodal redundancy and the high computational cost of processing visual elements remains a critical next step. Second, the effectiveness of our Reinforcement Learning stage is partially tied to the quality of the SFT initialization; in extremely niche domains where the base model fails to discover a single Minimal Necessary Path, the AEA mechanism lacks the necessary empirical anchors to bootstrap optimization. Finally, our current implementation employs a uniform weighting scheme for tool calls, which does not account for the varying latency and financial costs associated with different tools in real-world Deep Research deployments. Future work should investigate fine-grained, cost-sensitive tool weighting to better align efficiency rewards with practical operational constraints.

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning, 2025. *URL https://arxiv.org/abs/2503.04697*, 2(3).

anthropic. 2025. Introducing claude 4.

Song Bai, Lidong Bing, Carson Chen, Guanzheng Chen, Yuntao Chen, Zhe Chen, Ziyi Chen, Jifeng Dai, Xuan Dong, Wenhan Dou, Yue Deng, Yunjie Fu, Junqi Ge, Chenxia Han, Tammy Huang, Zhenhang Huang, Jerry Jiao, Shilei Jiang, Tianyu Jiao, and 35 others. 2025a. Mirothinker: Pushing the performance boundaries of open-source research agents via model, context, and interactive scaling. *Preprint*, arXiv:2511.11793.

Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, and 149 others. 2025b. Kimi k2: Open agentic intelligence. *Preprint*, arXiv:2507.20534.

Zheng Chu, Xiao Wang, Jack Hong, Huiming Fan, Yuqi Huang, Yue Yang, Guohai Xu, Chenxiao Zhao, Cheng Xiang, Shengchao Hu, and 1 others. 2026. Redsearcher: A scalable and cost-efficient framework for long-horizon search agents. *arXiv preprint arXiv:2602.14234*.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Run-Ze Fan, Zengzhi Wang, and Pengfei Liu. 2025. Megascience: Pushing the frontiers of post-training datasets for science reasoning. *arXiv preprint arXiv:2507.16812*.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. 2025. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous rl. *arXiv preprint arXiv:2508.07976*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dongyu Chen. 2024. Webvoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

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

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, Jialong Wu, Jingren Zhou, Kuan Li, Liangcai Su, Litu Ou, Liwen Zhang, Pengjun Xie, Rui Ye, Wenbiao Yin, Xinmiao Yu, Xinyu Wang, Xixi Wu, and 37 others. 2025a. Tongyi deepresearch technical report. *Preprint*, arXiv:2510.24701.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, and 1 others. 2025b.

Websailor: Navigating super-human reasoning for web agent. *arXiv preprint arXiv:2507.02592*.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yongkang Wu, Ji-Rong Wen, Yutao Zhu, and Zhicheng Dou. 2025c. Webthinker: Empowering large reasoning models with deep research capability. *Preprint*, arXiv:2504.21776.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yongkang Wu, Ji-Rong Wen, Yutao Zhu, and Zhicheng Dou. 2025d. Webthinker: Empowering large reasoning models with deep research capability. *arXiv preprint arXiv:2504.21776*.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. 2025e. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*.

Junteng Liu, Yunji Li, Chi Zhang, Jingyang Li, Aili Chen, Ke Ji, Weiyu Cheng, Zijia Wu, Chengyu Du, Qidi Xu, Jiayuan Song, Zhengmao Zhu, Wenhu Chen, Pengyu Zhao, and Junxian He. 2025. Webexplorer: Explore and evolve for training long-horizon web agents. *Preprint*, arXiv:2509.06501.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. Cot-valve: Length-compressible chain-of-thought tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6025–6035.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*.

Tergel Munkhbat, Namgyu Ho, Seo Hyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. 2025. Self-training elicits concise reasoning in large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 25127–25152.

OpenAI. 2025a. Deep research system card.

OpenAI. 2025b. Introducing openai o3 and o4-mini.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, and 1 others. 2025. Humanity's last exam. *arXiv preprint arXiv:2501.14249*.

Evan F Risko and Sam J Gilbert. 2016. Cognitive offloading. *Trends in cognitive sciences*, 20(9):676–688.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Dingfeng Shi, Jingyi Cao, Qianben Chen, Weichen Sun, Weizhen Li, Hongxuan Lu, Fangchen Dong, Tianrui Qin, King Zhu, Minghao Liu, and 1 others. 2025. Taskcraft: Automated generation of agentic tasks. *arXiv preprint arXiv:2506.10055*.

Sijun Tan, Michael Luo, Colin Cai, Tarun Venkat, Kyle Montgomery, Aaron Hao, Tianhao Wu, Arnav Balyan, Manan Roongta, Chenguang Wang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. rllm: A framework for post-training language agents. https://pretty-radio-b75.notion.site/rLLM-A-Framework-for-Post-Training-Language-Agents-21b8 Notion Blog.

Zhengwei Tao, Haiyang Shen, Baixuan Li, Wenbiao Yin, Jialong Wu, Kuan Li, Zhongwang Zhang, Huifeng Yin, Rui Ye, Liwen Zhang, Xinyu Wang, Pengjun Xie, Jingren Zhou, and Yong Jiang. 2025. Webleaper: Empowering efficiency and efficacy in webagent via enabling info-rich seeking. *Preprint*, arXiv:2510.24697.

Zhengwei Tao, Jialong Wu, Wenbiao Yin, Junkai Zhang, Baixuan Li, Haiyang Shen, Kuan Li, Liwen Zhang, Xinyu Wang, Yong Jiang, and 1 others. Webshaper: Agentically data synthesizing via information-seeking formalization, 2025. *URL https://arxiv. org/abs/2507.15061*.

Hongru Wang, Boyang Xue, Baohang Zhou, Tianhua Zhang, Cunxiang Wang, Huimin Wang, Guanhua Chen, and Kam-Fai Wong. 2025. Self-dc: When to reason and when to act? self divide-and-conquer for compositional unknown questions. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6510–6525.

Junjie Wang, Zequn Xie, Dan Yang, Jie Feng, Yue Shen, Duolin Sun, Meixiu Long, Yihan Jiao, Zhehao Tan, Jian Wang, Peng Wei, and Jinjie Gu. 2026. Webclipper: Efficient evolution of web agents with graph-based trajectory pruning. *Preprint*, arXiv:2602.12852.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. 2025. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*.

et al. Wu. 2024. Dler: Doing length penalty right - incentivizing more intelligence per token via reinforcement learning. *arXiv preprint*.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Gang Fu, Yong Jiang, and 1 others. 2025a. Webdancer: Towards autonomous information seeking agency. *arXiv preprint arXiv:2505.22648*.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He,

Deyu Zhou, Pengjun Xie, and 1 others. 2025b. Webwalker: Benchmarking llms in web traversal. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10290–10305.

Xbench Team. 2025. Xbench-deepsearch.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of draft: Thinking faster by writing less, 2025. *URL https://arxiv. org/abs/2502.18600*.

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, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, Yean Cheng, and 151 others. 2025. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *Preprint*, arXiv:2508.06471.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, and 1 others. 2024. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. 2024. Swift:a scalable lightweight infrastructure for fine-tuning. *Preprint*, arXiv:2408.05517.

A Implementation Details

A.1 Data Preparation

To construct a robust and high-signal training environment for **SlimSearcher**, we curated specialized datasets for both the SFT and RL stages. These high-quality training instances were meticulously curated through iterative rejection sampling and answer consistency verification, drawn from a diverse collection of information-seeking datasets, including Asearcher (Gao et al., 2025), TaskCraft (Shi et al., 2025), WebWalker (Wu et al., 2025b), Voyager (He et al., 2024), WebShaper (Tao et al.), REDSearcher (Chu et al., 2026), WebDancer (Wu et al., 2025a), and MegaScience (Fan et al., 2025). Furthermore, we synthesized a series of additional training instances by building upon the synthetic data generation paradigms introduced by WebSailor (Li et al., 2025b) and WebDancer (Li et al., 2025b). We present representative cases of our synthesized data in Figure 4. Our data preparation pipeline prioritizes complex, long-horizon web navigation tasks while strictly enforcing efficiency constraints. **All data will be made publicly available upon acceptance.**

A.2 Details of Rejection Sampling

To construct a high-quality distillation corpus, we aggregate several public QA datasets alongside partially synthetic data. To ensure the complexity and quality of training signals, we execute each query four times within the *Tongyi-DeepResearch* environment. We specifically retain only samples exhibiting a pass rate within the interval $0 < \mathrm{PR}(q) \leq 1$. This filtering criterion targets "challenging yet solvable" tasks, effectively excluding trivial queries that do not require deep research and impossible tasks that might introduce noise. The resulting filtered trajectories serve as the foundation for the **SFT Stage**. Unlike traditional rejection sampling, which typically selects a successful trajectory at random from the candidate pool, our framework employs **Pareto-efficient filtration**. Among all correct candidate trajectories $\{\tau_1, \ldots, \tau_K\}$ for a given query, we identify and distill the *Minimal Necessary Path* (MNP). We rank these candidates by their joint efficiency score (derived from the weighted tool-call costs and token counts defined in Section 3.1) and select only the top-performing trajectory for fine-tuning.

A.3 SFT stage

For the SFT stage, we utilized a carefully curated dataset comprising 13,863 training trajectories sourced from Asearcher (Gao et al., 2025), TaskCraft (Shi et al., 2025), WebWalker (Wu et al., 2025b), and partially synthetic data filtered via our proposed method (Section 3.2). The training is conduct on the ms-swift framework (Zhao et al., 2024). We implemented a sophisticated two-phase learning rate schedule, beginning with a 5% warmup period during which the learning rate linearly increases from 1×10^{-10} to 5×10^{-6}, followed by a 95% cosine decay phase that gradually reduces the learning rate from 5×10^{-6} back to 1×10^{-10} to ensure stable convergence. Training was conducted with a micro-batch size of 1 per GPU and a global batch size of 16, leveraging an advanced **four-level parallelism strategy**: expert model parallelism (2-way), tensor model parallelism (2-way), pipeline model parallelism (8-way), and context parallelism (2-way). This configuration requires a minimum of 64 GPUs across 8 computational nodes to achieve optimal data parallelism. This hybrid parallel architecture enables efficient processing of 128K context lengths, with gradient accumulation over 16 steps to reach the target global batch size.

A.4 RL stage

To facilitate robust adversarial evolution, we constructed a specialized environment for the RL stage. Our curated training corpus comprises 1,510 high-quality QA pairs sampled from a diverse collection of information-seeking datasets: Voyager (He et al., 2024), WebShaper (Li et al., 2025b), REDSearcher (Chu et al., 2026), WebDancer (Wu et al., 2025a), and MegaScience (Fan et al., 2025). The RL in conduct on the RLLM framework (Tan et al., 2025). We employ the **GRPO** algorithm (Shao et al., 2024) with a training batch size of 64 and a validation batch size of 128. The prompt length is capped at 8,000 tokens, while the maximum response length is set to 120,000 tokens to accommodate long-horizon reasoning trajectories. The actor model is optimized with a learning rate of 1×10^{-6}, using a PPO mini-batch size of 32 and dynamic batch sizing with a maximum token length of 16,000 tokens per GPU.

For rollout, we utilize **vLLM** (Kwon et al., 2023) in asynchronous mode with a tensor model parallel size of 8 and a GPU memory utilization of 0.3. During training, 8 responses are sampled per prompt at

Question:
Who propounded the concept of the rule of law in a nation whose diplomatic recognition of a Middle Eastern state—contrary to its closest ally's position—was welcomed by a regional political organization whose legislative body allocates exactly four seats per member, which recently reinstated a country's membership after a decade-long suspension, and whose port serves as a key European stop on a new Arctic shipping route originating from a coastal province which, serving as a junction with both a province ranking second in the nation for its number of sites belonging to a heritage designation that, established in the mid-2010s by an international commission, requires candidate projects to have operated for over a century and whose registry is numerically dominated by entries from a single Asian nation, where the rural tap water penetration rate has exceeded ninety-four percent, and a province where residents have the highest surveyed rate of traditional medicine consumption, even as a separate multi-year analysis shows foreign pharmaceuticals leading sales in its medical institutions, and housing a world-leading battery producer, registers a point-out degree centrality of exactly three in a national study of medical device innovation networks?

Answer:
Albert Venn Dicey

Question:
Between the animation duo—a partnership from a country where, for the first time in over three decades, its two most storied universities failed to place in the top three of a major national ranking while a South China business school was recognized as an academic institution whose diverse partnerships include a third-year student exchange with a university's Weihai campus, a dual-degree program with a Guangdong business school, and a contribution to an international medical consensus statement on aging—and the home nation of a tech giant that established its inaugural AI center in a region whose conflict-related mental health crises are the subject of academic study, while simultaneously becoming a scholarship-funded higher education destination for students from an East Asian nation turning away from Western universities, a country whose national oncological research body is a listed participant in expert reviews by a Swiss-headquartered intergovernmental organization that established a global day for patient safety in the year immediately preceding its declaration of the novel coronavirus as a public health emergency of international concern—whose aesthetic is cited as a key influence on a Czech animator, and the director of a 2015 documentary about autistic youths preparing for a formal dance in an Ohio city, who was the first to win an award?

Answer:
Brothers Quay

Figure 4: Cases of our synthesized data.

temperature 1.0, while validation decoding adopts temperature 0.7, top-p of 0.8, and top-k of 20. Both actor and reference model FSDP configurations enable parameter offloading, and gradient checkpointing is activated to reduce memory overhead. Ulysses sequence parallelism of size 8 is applied to the actor for efficient long-context training.

The entire RL training pipeline is deployed on 64 H800 GPUs across 8 nodes. Each agent is allowed a maximum of 100 environment interaction steps with a trajectory timeout of 7,200 seconds. Training proceeds for up to 10 epochs, with checkpoints saved every 10 steps.

A.5 Datasets

To comprehensively evaluate the proposed framework, we conduct experiments on four widely adopted web agent benchmarks: XBench-DeepSearch (Xbench Team, 2025), BrowseComp (Wei et al., 2025), GAIA (Mialon et al., 2023), and HLE (Phan et al., 2025). For the GAIA benchmark, we utilize a subset of 103 text-only queries from the development set. For the HLE benchmark, following the experimental protocol of previous studies (Li et al., 2025c), we evaluate the models on a subset of 500 text-only questions.

A.6 Evaluation Details

For GAIA and xBench, following the evaluation protocol of Li et al. (2025a), we adopt Qwen2.5-72B-Instruct as the judge model. The evaluation prompt remains identical to that used in their work to ensure consistency and comparability. For xBench-DeepSearch, we adopt Gemini-2.5-Flash as the judge model. For BrowseComp, we employ GPT-4o-2024-11-20 as the judge model. For Humanity's Last Exam, we evaluate the 500 text-only questions following (Li et al., 2025d). The evaluation prompt follows the official protocol, with GPT-4o-2024-11-20 serving as the evaluator. The evaluation prompts for all benchmarks are kept consistent with those described in the respective original pa-

pers to ensure alignment and reproducibility. The detailed evaluation prompts for each benchmark will be made publicly available on our GitHub repository upon acceptance.

B System Prompt

In this section, we present the complete system prompt used to initialize the **SlimSearcher** agent. This prompt serves as the foundational instruction set, defining the agent's core identity as a "Deep Research Assistant." It explicitly outlines the available external tools—including web search, page visiting, Python code execution, and file parsing—along with their precise input parameters. Furthermore, it enforces a strict interaction schema, requiring the agent to formulate tool invocations as standardized JSON objects within specific XML tags and to encapsulate its final conclusion within `<answer>` tags. The exact prompt template is illustrated in Figure 5.

C Case Study

To qualitatively evaluate **SlimSearcher's** capability in circumventing the "efficiency trap" inherent in vanilla base models, we conduct a detailed analysis comparing reasoning traces against the baseline agent, MiroThinker. We present three representative examples from GAIA and XBench, visualized in Figures 6 ,7 and 8.

System Prompt

You are a deep research assistant. Your core function is to conduct thorough, multi-source investigations into any topic. You must handle both broad, open-domain inquiries and queries within specialized academic fields. For every request, synthesize information from credible, diverse sources to deliver a comprehensive, accurate, and objective response. When you have gathered sufficient information and are ready to provide the definitive response, you must enclose the entire final answer within **<answer></answer>** tags.

 # Tools

You may call one or more functions to assist with the user query. You are provided with function signatures within **<tools></tools>** XML tags:

<tools>

{"type": "function", "function": {"name": "**search**", "description": "Perform Google web searches then returns a string of the top search results. Accepts multiple queries.", "parameters": {"type": "object", "properties": {"query": {"type": "array", "items": {"type": "string", "description": "The search query."}, "minItems": 1, "description": "The list of search queries."}}, "required": ["query"]}}}

{"type": "function", "function": {"name": "**visit**", "description": "Visit webpage(s) and return the summary of the content.", "parameters": {"type": "object", "properties": {"url": {"type": "array", "items": {"type": "string"}, "description": "The URL(s) of the webpage(s) to visit."}, "goal": {"type": "string", "description": "The specific information goal for visiting webpage(s)."}}, "required": ["url", "goal"]}}}

{"type": "function", "function": {"name": "**PythonInterpreter**", "description": "Executes Python code in a sandboxed environment. To use this tool: 1. The `arguments` JSON object must be empty: {}. 2. The Python code must be placed after the JSON block, enclosed within **<code></code>** tags. Any output MUST be printed using `print()`. Example: **<tool_call>** {"name": "PythonInterpreter", "arguments": {}} **<code>** import numpy as np # Your code here print(f"The result is: {np.mean([1,2,3])}") **</code> </tool_call>**", "parameters": {"type": "object", "properties": {}, "required": []}}}

{"type": "function", "function": {"name": "**google_scholar**", "description": "Leverage Google Scholar to retrieve relevant information from academic publications. Accepts multiple queries. This tool will also return results from Google search.", "parameters": {"type": "object", "properties": {"query": {"type": "array", "items": {"type": "string", "description": "The search query."}, "minItems": 1, "description": "The list of search queries for Google Scholar."}}, "required": ["query"]}}}

{"type": "function", "function": {"name": "**parse_file**", "description": "This is a tool that can be used to parse multiple user uploaded local files such as PDF, DOCX, PPTX, TXT, CSV, XLSX, DOC, ZIP, MP4, MP3.", "parameters": {"type": "object", "properties": {"files": {"type": "array", "items": {"type": "string"}, "description": "The file name of the user uploaded local files to be parsed."}}, "required": ["files"]}}}

</tools>

For each function call, return a json object with function name and arguments within **<tool_call></tool_call>** XML tags:

<tool_call>

{"name": <function-name>, "arguments": <args-json-object>}

</tool_call>

Current date:

Figure 5: The complete system prompt used to initialize the SlimSearcher agent.

SlimSearcher vs. MiroThinker

Question

The work referenced in footnote 397 of Federico Lauria's 2014 dissertation is also the source for the titles of two paintings in the Smithsonian American Art Museum's collection. What is the absolute difference between the chapter numbers referenced in the titles of these two paintings?

SlimSearcher (Ours)

Input: Q **Output:** A

Phase 1: Locate Dissertation
1. `search`(*"Lauria 2014 dissertation"*) $\rightarrow$ thesis PDF
2. `visit`(PDF) $\rightarrow$ download to sandbox

Phase 2: Extract Footnote
3. `parse_pdf`() $\rightarrow$ search "397"
4. `extract`(fn. 397) $\rightarrow$ *Hobbes, Leviathan*
5. `verify`() $\rightarrow$ work = *Leviathan* ✓

Phase 3: Find Paintings
6. `search`(*"SAAM Leviathan paintings"*) $\rightarrow$ collection page
7. `visit`(SAAM) $\rightarrow$ extract list
8. `filter`() $\rightarrow$ P_1: Ch. 15, P_2: Ch. 23

Phase 4: Calculate
9. $A \leftarrow |15 - 23| = 8$; `verify`() $\rightarrow$ stop

Tools: **22** | Searches: **15** **Return** $A = \mathbf{8}$ ✓

MiroThinker

Input: Q **Output:** A

Loop 1: Search Dissertation (tools 1–45)
for $i = 1$ **to** 8: `search`("Lauria" variant i)
for $j = 1$ **to** 12: `scrape`(); **if** uncertain: re-scrape ▷ *low conf.*
for $k = 1$ **to** 10: `search`("footnote 397" variant k)

Loop 2: Extract Reference (46–95)
`create_sandbox`(); `download_pdf`()
for $m = 1$ **to** 15: `parse_pdf`(method m); `search`("397")
for $n = 1$ **to** 10: `verify`(); **if** uncertain: re-verify ▷ *loop trap*

Loop 3: Find Paintings (96–200)
for $p = 1$ **to** 25: `search`("SAAM Leviathan" variant p)
for $q = 1$ **to** 40: `scrape`(SAAM); `extract`(list)
for $r = 1$ **to** 20: `verify`(titles)

Loop 4: Final Verification (201–288)
for $s = 1$ **to** 30: `search`(verify chapters)
for $t = 1$ **to** 30: `calculate`($|c_1 - c_2|$); `verify`()
for $u = 1$ **to** 28: `final_check`()

Tools: **288** | Searches: **134** **Return** $A = \mathbf{8}$ ✓

SlimSearcher: 22 tools, 15 searches **vs. MiroThinker**: 288 tools, 134 searches

Figure 6: Reasoning trace comparison on GAIA. SlimSearcher solves in 4 phases with 22 tools; MiroThinker uses $13\times$ more tools across 4 redundant loops yet reaches the same answer.

Algorithm Comparison: SlimSearcher vs. MiroThinker on XBench

Question

黑龙江、吉林、辽宁，共有多少个地市级行政单位与外国接壤?

SlimSearcher (Ours)

Input: Question Q **Output:** Answer A

1. `search`(“黑龙江吉林辽宁边境城市”)
→ find comprehensive list for all 3 provinces

2. `visit`(*Wikipedia: border cities*)
→ extract structured list of all border cities

3. `scrape`(*verify count*)
→ confirm total count from table

4. Aggregate results
$A \leftarrow$ count all border cities (12 total)

Return $A = \mathbf{12}$ ✓

Tools: **3** | Searches: **1** | Visits: **2**

MiroThinker

Input: Question Q **Output:** Answer A

```
for province p ∈ {黑, 吉, 辽} do
  for city c in p do
    search("city c border Russia")
    visit(Wikipedia page for c)
    if uncertain then
      search("city c border verification")
      scrape(verify again)    ▷ loop trap
    end if
  end for
end for
```

Reached 400-turn limit without completing...

Return *No answer* ×

Tools: **403** | Searches: **359**

SlimSearcher: 3 tools → **Correct (12)** **vs.** **MiroThinker**: 403 tools → **Failed**

Figure 7: Comparison of reasoning traces on XBench . SlimSearcher uses a strategic aggregation approach, while MiroThinker falls into an exhaustive verification loop.

Algorithm Comparison: SlimSearcher vs. MiroThinker on XBench

Question

有一种美食，起源于中国南北朝时期，分为两大流派，可以裹饼吃、也可以蘸糖吃，这个美食是什么?

MiroThinker

Input: Question Q **Output:** Answer A

1–20. `search`(“南北朝美食”)
⋮ *20 query variants*

21–50. `search`(“烤鸭历史起源”)
`scrape`(verify 南北朝)
⋮ ▷ *redundant verification*

51–102. `search`(“全聚德烤鸭”)
`search`(“荷叶饼裹鸭”)
⋮ ▷ *over-verification loop*

Return A = 北京烤鸭 ✓

Tools: **102** | Searches: **51** | Visit: **51**

SlimSearcher (Ours)

Input: Question Q **Output:** Answer A

1. `search`(“南北朝美食两大流派裹饼蘸糖”)
→ locate comprehensive food encyclopedia

2. `Visit`(*food history page*)
→ extract: 北京烤鸭(Peking Duck)
→ verify: 挂炉vs 焖炉(two schools)

Return A = 北京烤鸭 ✓

Tools: **2** | Searches: **1** | Visit: **1**

SlimSearcher: 2 tools **vs.** **MiroThinker**: 102 tools ⇒ **51× overhead**

Figure 8: Comparison of reasoning traces on XBench.