Fathom-Search-4B: Unlocking Long-Horizon DeepSearch for SLMs

Shreyas Singh*
Fractal AI Research
shreyas.singh@fractal.ai

Pradeep Moturi*
Fractal AI Research
pradeep.moturi@fractal.ai

Kunal Singh*†
Fractal AI Research
kunal.singh@fractal.ai

Abstract

Tool-integrated reasoning has emerged as the key focus for enabling agentic applications . Among them, DeepResearch Agents have received significant attention due to impressive performance on complex information seeking tasks. We present Fathom-Search-4B, tool-using LLM specialized in evidence-based DeepSearch. Our approach combines three advances. First, DUETQA, a 5K-sample training dataset generated via our novel multi-agent self-play framework that can be used to synthesize question–answer pairs with strict live-web-search dependence, post model cut-off date bias, and heterogeneity of web sources beyond Wikipedia. Second, we introduce RAPO, a zero-overhead extension of GRPO that stabilizes multi-turn Reinforcement Learning with Verifiable Rewards (RLVR) via three upgrades: (i) curriculum-inspired pruning of saturated prompts; (ii) reward-aware advantage scaling that preserves gradient magnitude under sparse rewards; and (iii) a per-prompt replay buffer that injects the latest successful rollout into failed groups, restoring reward variance and stabilizing relative-advantage estimates. Third, we design a steerable step-level reward that classifies each tool call by cognitive role and marginal utility (e.g., exploration, verification, redundancy), enabling explicit control over search breadth, cross-source verification depth, and overall tool-use horizon; this reliably extends effective trajectories beyond 20 tool calls when warranted. The agent operates with a goal-conditioned web-search stack (live web search via a search engine + targeted web-page querying via an LLM). Evaluated on DeepSearch benchmarks (e.g., SimpleQA, FRAMES, WebWalker, Seal0, MuSiQue) and out-of-domain reasoning suites (HLE, AIME-25, GPQA-Diamond, MedQA), Fathom-Search-4B attains state-of-the-art results in the open-weights category across all DeepSearch benchmarks, and achieves significant improvements in general reasoning tasks via tool-integrated reasoning.

1 Introduction

Large Language Models (LLMs) have demonstrated promising results across a diverse set of tasks, such as mathematical reasoning, code generation [10, 5, 25, 24]. Despite these advancements, they remain prone to factual inaccuracies/hallucinations as they rely on static internal knowledge acquired

* Equal contribution
† Project Lead.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Foundations of Reasoning in Language Models.

during pretraining. Real world information is continually evolving and getting updated. Given the high cost of pretraining LLMs, it is not pragmatic to rely solely on repeated pretraining to update their knowledge. A potential solution to this problem involves enabling LLMs to interface with external knowledge systems.

Retrieval-augmented generation (RAG) has become the standard framework for open-domain QA, where LLMs generate answers conditioned on retrieved context. However, these pipelines rely on structured, static corpora and predictable input formats conditions rarely met in real-world search tasks. In contrast, recent efforts [27, 14, 26] have focused on instilling tool-mediated DeepSearch reasoning capabilities in LLMs that involve free-form web queries, parsing heterogeneous/noisy outputs, and synthesizing multi-step reasoning chains. The core principle underlying this approach is to concentrate training efforts on developing the model's ability to autonomously and effectively access and leverage external information sources (search engines). Rather than directly incorporating factual content into the model parameters, this method emphasizes teaching the model how to navigate, comprehend, and utilize relevant information from vast digital information landscapes for complex-information seeking tasks. This distinction makes DeepSearch fundamentally more difficult than traditional RAG and a necessary precursor to the next scaling milestone in search augmented language model capabilities: *DeepResearch Agents*

However, scaling DeepSearch capability faces three key challenges: (i) the lack of high-quality, verifiable, scalable training dataset creation pipeline , (ii) algorithmic instability in multi-turn reinforcement learning (RL) with tools , and (iii) inefficient tool calling behavior which hinders scaling deep information exploration and retrieval capabilities

1.1 Motivation

(1) Training instability of GRPO in multi-turn tool interaction: RLVR (Reinforcement Learning with verifiable rewards) with GRPO [23] has demonstrated early promise in aligning LLMs with sparse reward signals for single-turn reasoning tasks, particularly in structured domains like Math/STEM **(author?)** [23, 35]. However, GRPO struggles to scale to multi-turn tool-augmented environments, because external tool interaction responses induce distribution shift in the policy model from its set token generation patterns, this leads to decoding instability and malformed generations. This cascading of errors causes group-relative advantages to saturate, leading to extremely unstable gradient updates that breaks the entire training process. [33].

(2) Reward hacking and inefficient tool calling *(a) Correctness-only sparse rewards do not scale to long-horizon tool calling.* When training with only a single end of episode correctness signal, the agent shows early improvements achieving format adherence and basic tool-calling competence in the beginning, however, as training progresses, tool usage increases sharply while both training reward and validation performance deteriorate [**?**]. This degradation stems from reward hacking: the agent collapses into repetitive, identical tool calls because the vanilla RLVR objective provides no incentive for efficiency or diversity in tool use. *(b) RL amplifies SFT priors, limiting control over the cognitive behaviors developed by the policy* [7]: Tool-use RL typically relies on an SFT cold start to elicit basic tool competence [14]; [6] RL then amplifies pre-existing cognitive behaviors seeded by SFT. Standard RLVR affords limited control over the exploration and verification strategies developed by the policy model, consequently the quality of cold-start trajectories disproportionately shape the policy model's tool-use behavior and provides no steerability.

(3) Limited training data characterized by high and hard-to-reduce intrinsic information uncertainty: Training datasets such as TriviaQA [12], and multi-hop variants like 2WIKI[8], and HotpotQA [36] represent problems where solutions can often be found through minimal set queries or even from a model's parametric knowledge alone. These datasets do not expose models to the real-world retrieval challenges posed by noisy, heterogeneous data sources on the internet. Recent synthetic efforts [27, 14, 28] attempt to bridge this gap by simulating realistic search behavior. For instance, WebSailor's[14] SailorFog-QA constructs ambiguous queries using obfuscated subgraphs of entity graphs, while SimpleDeepResearcher [28] issues multi-stage search-summarize-generate tool calls over raw HTML. Despite their innovation, these pipelines remain expensive, brittle, and time-consuming. They rely on handcrafted heuristics, graph expansion, or multi-stage LLM orchestration, limiting scalability, topical diversity, and adaptability to new domains.

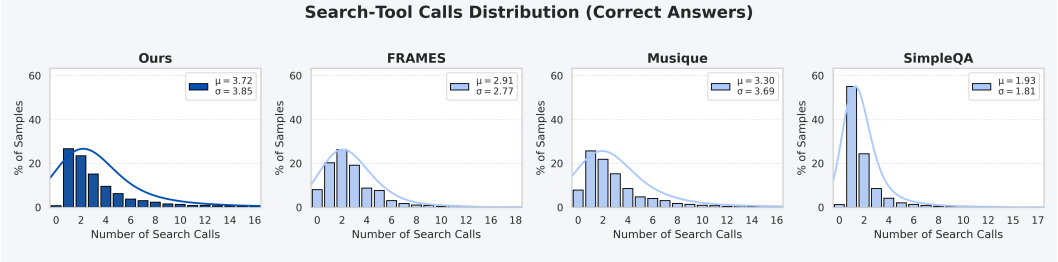

Figure 1: Distribution of search-call counts issued by o3 over correctly answered questions per benchmark, comparing DuetQA with other prominent benchmarks. DuetQA shows strict live-web-search dependence: o3[18] solves close to 0 questions DuetQA items without issuing a search call.

1.2 Our Contributions

To this end, we introduce a post-training recipe to create state-of-the-art DeepSearch enabled reasoning model, Fathom-Search-4B. We enlist our key contributions below:

- **Multi-agent self-play dataset.** We build DUETQA, a 5K-sample dataset created through multi-agent self-play, that are impossible to answer without *live web search*.
- **Two-stage RL-zero training.** We introduce a two-stage RL-Zero training framework that provides coarse control over the cogitive behaviors developed by the policy model with regards to exploration and verification strategies.
- **RAPO** We propose RAPO, a zero-overhead modification of GRPO that stabilizes multi-turn RL through dataset pruning, advantage scaling, and replay buffers.
- **Steerable step-level reward.** We design a novel step level reward that enables fine-grained control over long-horizon tool use, which scales tool use beyond $20+$ calls

2 Methodology

We describe the methodology underlying *Fathom-Search*, a tool-using LLM that leverages live web-search capabilities to do evidence based reasoning in a multi-turn tool interaction setting, achieving long-horizon tool use (>20 calls) when warranted. These capabilities arise from a combined approach of: (i) a curated synthetic data pipeline tailored to search-tool augmented reasoning, (ii) targeted upgrades to GRPO to effectively adapt it to multi-turn tool interaction, and (iii) a two-stage training regimen with reward shaping to expand the tool-use horizon in a steerable manner.

2.1 DuetQA: A synthetic Deep-search dataset, generated via multi-agent self play

To address the aforementioned challenges in Section 1., we develop a self-supervised dataset construction framework designed to yield verifiable, search-dependent, multi-hop QA pairs. This pipeline serves as the basis for generating DUETQA, a dataset tailored for training agentic deepsearch models. The design goals are: **Live web-search dependency**: for each QA pair (q, a), the question is unanswerable without search by enforcing that at least one hop contains information post–2024-01-01 (i.e., for a model $\mathcal{M}$, $P(a \mid q, \mathcal{M}_{\text{no-search}}) \ll P(a \mid q, \mathcal{M}_{\text{search}})$); **Diverse source domains**: questions require querying beyond Wikipedia (e.g., online PDFs, news outlets, government filings, academic blogs, discussion forums); and **Steerable theme control**: each example is grounded in $k \in [5, 7]$ sampled themes $\mathcal{T}_{\text{sample}} \subset \mathcal{T}$, where $\mathcal{T}$ is a manually curated taxonomy of 200+ themes covering a broad range of topics. We generate questions using two frontier web search enabled LRMs, $\mathcal{M}_1$ (O3) and $\mathcal{M}_2$ (O4-mini) [18], acting as *proxy web-crawling agents* that produce QA pairs and as *independent verifiers* to that ensure question solvability; a third model, $\mathcal{M}_3$ (GPT-4o), is a *non-search* model used for controlled paraphrasing/obfuscation of questions and as a baseline verifier without search.

Mixture of Themes mode. To ensure thematic diversity in the generated question while also guaranteeing recency and search dependence, we sample $\mathcal{T}_{\text{sample}} \sim \text{Uniform}(\mathcal{T})$ with $|\mathcal{T}_{\text{sample}}| = k$,

$k \in \{5, 6, 7\}$. For each $t \in \mathcal{T}_{\text{sample}}$, the generator (either $\mathcal{M}_1$ or $\mathcal{M}_2$) issues live queries to retrieve recent and/or obscure facts, with the constraint that at least one fact references information post–2024-01-01. The generator then composes a multi-hop question q by logically chaining the k facts—one hop per theme—into a coherent reasoning path

Seeded Question mode. To approximate real-world query distribution and logical chaining patterns observed in hard multi-hop questions, we construct a seed bank of 100 questions (50% manually authored; 50% from **BrowseComp** [31]). For each seed q_0, we sample candidate themes $\{t_1, \ldots, t_k\} \subset \mathcal{T}$ with $k \in \{3, 4, 5\}$ and rewrite q_0 into a new question q by integrating one or more sampled themes that satisfy the obscurity/recency constraints while preserving the seed's multi-hop scaffold / logical chaining patterns.

Data obfuscation To remove surface cues that let models *short-circuit* the intended multi-hop reasoning, we apply a dedicated obfuscation pass after question generation. Using the non-search model $\mathcal{M}_3$ (GPT-4o) under an in-context learning setup with exemplars, we paraphrase the question to mask intermediate hops. Concretely, $\mathcal{M}_3$ softens exact anchors in each hop by (i) converting specific dates to coarse intervals ("March 2025" $\rightarrow$ "early 2025"), (ii) mapping precise numerics to qualitative magnitudes ("1%" $\rightarrow$ "negligible"), (iii) replacing named entities with indirect descriptors ("University of Florida" $\rightarrow$ "a major southeastern university"), and (iv) embedding causal/comparative pivots as descriptors rather than explicit connectors. These edits suppress shortcut signals without altering the underlying facts that must be recovered via search.

Multi-agent Verification After the obfuscation pass, we validate that each QA pair remains both *answerable* and *search-dependent*. We retain (q, a) only if the two search-enabled LRMs are able to answer the question correctly while a strong non-search baseline fails, i.e., $\mathcal{M}_1^{\text{search}}(q) = \mathcal{M}_2^{\text{search}}(q) = a \neq \mathcal{M}_3^{\text{no-search}}(q)$. This post-obfuscation check enforces correctness via cross-model agreement and certifies that web retrieval is necessary; it also filters cases where paraphrasing either leaked the answer or inadvertently made the item unsolvable.

2.2 Agentic Reinforcement Learning

In this section, we formulate multi-turn, tool-augmented RL with LLM policies. Let $x \in \mathcal{X}$ be an input sampled from a data distribution $\mathcal{D}$, and let $\mathcal{T}$ denote the set of available tools. The policy LLM π_θ interacts with tools to produce a *reasoning trajectory* $\mathcal{R}$ interleaved with tool-call feedback, followed by a final textual answer y. We include a reference policy π_{ref} for KL regularization, a verifiable reward function r_ϕ (parameterized LLM as judge), and a KL weight $\beta > 0$. We optimize a KL-regularized expected reward:

$$\max_{\pi_\theta} \; \mathbb{E}_{x \sim \mathcal{D},\, (\mathcal{R}, y) \sim \pi_\theta(\cdot \mid x; \mathcal{T})} \Big[r_\phi(x, \mathcal{R}, y) \Big] \; - \; \beta \, \mathbb{D}_{\text{KL}}[\pi_\theta(\mathcal{R}, y \mid x; \mathcal{T}) \parallel \pi_{\text{ref}}(\mathcal{R}, y \mid x; \mathcal{T})] \,. \quad (1)$$

Unlike conventional RL over pure text rollouts, agentic RL interleaves tool feedback into the *reasoning* process. We decompose the joint sampling as where $\mathcal{R} = \{\mathcal{R}_t\}_{t=1}^{t_\mathcal{R}}$ is the reasoning trajectory of length $t_\mathcal{R}$, interleaved with tool-call responses, and $y = \{y_t\}_{t=1}^{t_y}$ is the final answer of length t_y. Each reasoning step t can be viewed as a tuple

$$P_\theta(\mathcal{R}, y \mid x; \mathcal{T}) = \Big[\prod_{t=1}^{t_\mathcal{R}} P_\theta(\mathcal{R}_t \mid \mathcal{R}_{<t}, x; \mathcal{T}) \Big] \cdot \Big[\prod_{t=1}^{t_y} P_\theta(y_t \mid y_{<t}, \mathcal{R}, x; \mathcal{T}) \Big], \quad \mathcal{R}_t = (\varphi_t, c_t, o_t). \quad (2)$$

where φ_t is a latent "`think`" segment generated by the policy model enclosed in within the `<think></think>`, $c_t \in \mathcal{T}$ represents the chosen tool and its arguments enclosed within `<tool_call></tool_call>` tags and o_t is the "`response`" returned by the environment enclosed in the `<tool_response></tool_response>`, based on the ReAct template.

Optimization via GRPO. In practice, we optimize (1) with a token-level clipped surrogate. For a prompt-group of G sampled rollouts with scalar rewards $\{R_i\}_{i=1}^G$, we define group-relative advantages in (3)

$$\hat{A}_{i,t} \; = \; \frac{R_i - \mu_R}{\sigma_R}, \quad \mu_R = \tfrac{1}{G} \sum_{j=1}^{G} R_j, \quad \sigma_R = \sqrt{\tfrac{1}{G} \sum_{j=1}^{G} (R_j - \mu_R)^2}. \quad (3)$$

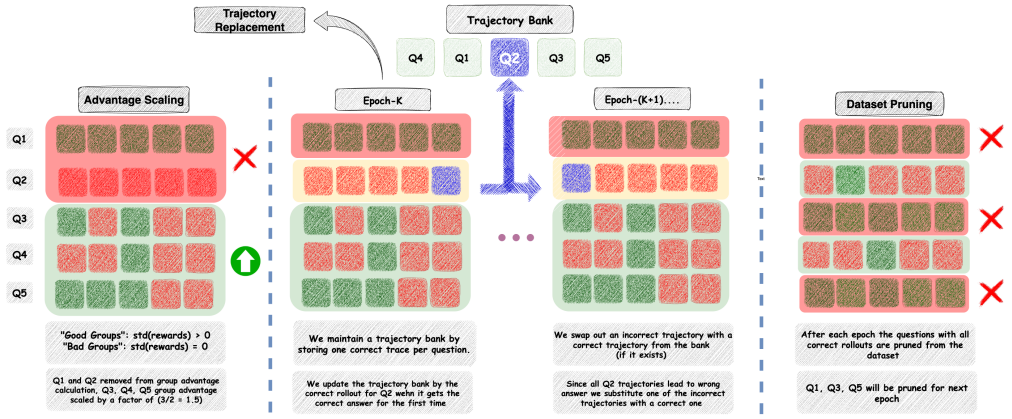

Figure 2: Visualzation of key ideas adapted by RAPO to stabalize GRPO training in multi-turn tool interaction scenarios. In-order (L-geR): i.) Advantage Scaling (Batch level), ii.) Trajectory Replacement(Group Level), & iii.) Dataset Pruning(Epoch Level)

and and minimize the clipped loss as defined in (4)

$$\mathcal{L}_{\text{GRPO}} = \frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{t=1}^{T_i}\min\Big[r_{i,t}\,\hat{A}_{i,t},\ \text{clip}\big(r_{i,t},1-\epsilon,1+\epsilon\big)\,\hat{A}_{i,t}\Big], \qquad r_{i,t} = \frac{\pi_\theta(o_{i,t}\mid x,\mathcal{H}_{t-1})}{\pi_{\theta_{\text{old}}}(o_{i,t}\mid x,\mathcal{H}_{t-1})}. \tag{4}$$

where the trajectory level reward in multi-turn RLVR setting is defined as follows

$$r_i \;=\; \alpha\, R_i^{\text{format}} \;+\; (1-\alpha)\, R_i^{\text{answer}}, \qquad \alpha \in [0,1] \tag{5}$$

Specifically, the format score R_i^{format} verifies whether the rollout trajectory follows the ReAct template; the answer score R_i^{answer} uses an LLM-as-judge to determine whether the final prediction matches ground truth.

2.3 Agentic Tool Design

We provide our policy model access to two tools for goal-conditioned web retrieval and reading. At step t, the agent emits $\mathcal{R}_t = (\varphi_t, c_t, o_t)$, where φ_t is latent `think` content, c_t is a tool call with arguments, and o_t is the tool response.

`search_urls` **(web search).** The tool takes as input a natural language query q and returns a ranked list of triples $(u, \texttt{title}, \texttt{snippet})$ using a live search engine. The policy uses this to identify promising sources and optionally select a URL u for opening in the next step. The tool is invoked as follows: `<tool_call>{name: search_urls, args: {query: q}}</tool_call>`

`query_url` **(goal-conditioned page reading).** Given a goal g and a URL u, the tool leverages a query LLM to return targeted evidence-backed response that address g. This tool enables precise grounding of facts and targeted querying of web-pages. Compared to the injestion of entire web-page into the policy model's trajectory, this tool minimizes noise and increases recall. The tool is invoked as follows: `<tool_call>{name: query_url, args: {goal: g, url: u}}</tool_call>`

2.4 RAPO: Reward Aware Policy Optimization

RAPO is a lightweight extension of GRPO that stabilizes multi-turn, tool-augmented training by mitigating the challeneges mentioned in Section. 1. Let σ_R be the within-group reward standard deviation from (3); a group is **Good** if $\sigma_R > 0$ and **Bad** if $\sigma_R = 0$. As Bad groups become more prevalent, either because all rollouts for a prompt converge to the correct answer (prompt saturation) or because multi-turn cascading of errors drive all rollouts to failure, the batch increasingly spends compute on rollouts that carry no advantage signal, causing the effective gradients to shrink, hence

destabilizing updates. RAPO applies three targeted modifications (on top of GRPO to counter precisely these effects with zero additional rollout cost.

Dataset pruning. We remove prompts at the end of every epoch that are effectively solved to avoid spending compute on groups that produce no training signal. This early-exit rule boosts throughput and implicitly induces a curriculum: as training progresses, the active set concentrates on harder prompts

$$\mathrm{SolveRate}(q) \;=\; \frac{1}{G}\sum_{i=1}^{G}\mathbf{1}[R_i>0], \qquad \text{prune if } \mathrm{SolveRate}(q)\geq 0.9 \tag{6}$$

Advantage scaling. When the proportion of Bad groups increase in a batch the magnitude of gradient norm decreases, as these Bad groups contribute no gradient signal. This causes the total gradient norm across the batch to collapse, causing ineffective and unstable training updates.To preserve the overall gradient magnitude, we reweight the token-level advantages of Good groups inversely by their batch frequency:

$$\tilde{A}_{i,t} \;=\; \frac{G}{G_{\text{good}}}\cdot \hat{A}_{i,t}, \qquad G_{\text{good}}=\#\{\sigma_R>0\}\,. \tag{7}$$

This preserves the magnitude of gradient updates without incurring additional rollout computation costs as done in DAPO, which effectively keeps regenerating trjectories untill it fills a batch with Good groups.

Replay buffer. To rescue Bad groups with all failed rollouts for a prompt, we keep a per-prompt buffer $\mathcal{B}$ with the most recent successful rollout $\mathbf{o}^\star$ (where $R(q,\mathbf{o}^\star)>0$). If the current group for q fully fails, we overwrite one of uniformly sampled rollout of the group with $\mathbf{o}^\star$:

$$\big(j\sim\mathrm{Uniform}\{1,\dots,G\},\ \mathbf{o}_j\leftarrow\mathbf{o}^\star\big). \tag{8}$$

This guarantees at least one successful trajectory in the group, restoring variance ($\sigma_R>0$) and enabling RAPO's relative advantage computation to penalize failed completions. Beyond gradient recovery, the successful trajectory also anchors the group with a high-quality, low-entropy reference, improving stability in the presence of distributional drift.

2.5 Steerable Step-Level Reward Design for Search Tools

We design a novel *Steerable Step-Level Reward* that alleviates the challenges faced by RLVR training in the multi-turn, tool-interaction setting using vanilla reward (5) as described in Section 1. The reward enables us to steer (i) *how much* the agent uses tools and (ii) *how* it allocates cognition to exploration and verification. Starting from the vanilla RLVR objective in (5), we make the correctness branch R_i^{answer} depend on novelty-aware labels assigned by a GPT-4.1 LLM-as-judge to each call c_t in $\mathcal{R}=\{(\varphi_t,c_t,o_t)\}_{t=1}^{T}$ as follows:

`search_urls` $\in\{$
UNIQUESEARCH: (semantically new query on unseen entities/facets),
REDUNDANTSEARCH: (near-duplicate of a prior query; overlapping results)$\}$

`query_url` $\in\{$
EXPLORATION: (first read of a new URL),
VERIFICATION: (cross-source check on a new URL for an existing query/fact; allowed B_v times),
REDUNDANTQUERY: (further checks for a query/fact on new URLs beyond B_v)$\}$

From the LLM-as-Judge tool call classificaiton we form tallies as follows:

$$n_{\text{uniqS}},\ n_{\text{redS}},\ n_{\text{explore}},\ n_{\text{verify}},\ n_{\text{redQ}}, \quad n_{\text{uniqQ}} \;=\; n_{\text{explore}}+n_{\text{verify}}, \quad T \;=\; |\mathcal{R}|\,. \tag{9}$$

and define the following aggregates:

$$\rho \;=\; \frac{n_{\text{redS}}+n_{\text{redQ}}}{T}, \qquad \Delta_S \;=\; n_{\text{uniqS}}-n_{\text{redS}}, \qquad \Delta_Q \;=\; n_{\text{uniqQ}}-n_{\text{redQ}}. \tag{10}$$

Using these summaries we define our *Steerable Step-Level Reward* as:

$$r_i \;=\; \begin{cases} \alpha\, R_i^{\text{format}} \;+\; \max\big((1-\alpha)(1-\rho),\, 0.5\big), & \text{if } R_i^{\text{answer}} = 1, \\ \alpha\, R_i^{\text{format}} \;+\; c_1\, \min\big(1,\, \frac{\Delta_S}{C_S}\big) \;+\; c_2\, \min\big(1,\, \frac{\Delta_Q}{C_Q}\big), & \text{if } R_i^{\text{answer}} = 0, \end{cases} \tag{11}$$

We set $\alpha = 0.1$ and $c_1 = c_2 = 0.2$ ($c_1{+}c_2 = 0.4$), to ensure any incorrect rollout has $r_i \leq 0.5$, while any correct rollout has $r_i \geq 0.5$, which ensures incorrect trajectories never get rewarded more than the correct ones. We set $c_1{=}c_2$, to allow equal weight to `search_urls` (Δ_S) and `query_url` (Δ_Q).

Steerability. We expose three primary knobs: (i) C_S and (ii) C_Q set the saturation thresholds for creditable novelty in `search_urls` and `query_url`, respectively. Increasing C_S and/or C_Q raises the novelty caps, enabling more steps to earn credit when they introduce genuinely new evidence; decreasing them compresses trajectories. (iii) The per-claim verification budget B_v controls verification depth: higher B_v permits multiple creditable cross-checks per claim, promoting verification. For our experiments we set B_v = 1 allowing 1 cross-check per claim, additionally we set C_S = 8 and C_Q = 16.

2.6 Training Recipe

We build our reinforcement learning with verifiable rewards (RLVR) framework on top of RECALL [3]. For web search, we use the Serper API [22], and implement a retrieval toolchain leveraging Jina-AI together with open-source components such as TRAFILTURA and CRAWL4AI. Training is carried out in two stages.

Stage 1. We train with RAPO for 10 epochs on our curated DUETQA dataset, comprising **4,988** high-quality QA instances. The setup uses a constant learning rate of 1×10^{-6} with the Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.95$), batch size 32, mini-batch size 16, 5 rollouts per group, and top-$p = 1.0$ sampling. Each rollout is capped at 32 tool-interaction steps, with each step limited to 8,192 output tokens. The vanilla reward (5) is used to instill correct tool-calling behavior and strict format adherence.

Stage 2. We continue RLVR training for an additional 2 epochs under the same hyperparameter settings. For Stage 2, we construct a mixed dataset by combining DUETQA, with math data from S1 dataset [16], and the training split of MUSIQUE [29]. This combined pool is adversarially filtered against the Stage-1 checkpoint, yielding **5,077** instances. From MUSIQUE, we retain only questions requiring at least three reasoning hops to ensure sufficient compositional depth. For this stage, we adopt the Steerable Step-Level Reward (11) to extend the tool-use horizon beyond 20 calls in a stable manner.

We use the Qwen3-4B model [34] as the base, which supports a maximum context length of 40,960 tokens; we utilize the full window during training. We use GPT-4.1-mini(Temperature=0.) as the query LLM for training and evaluation unless stated otherwsie. A higher sampling temperature of 1.4 is applied to Qwen3 models, consistent with prior findings [2]. All experiments are conducted on a single node with 8×H100 GPUs.

2.7 Baselines & Benchmarks

Baselines. Open-source DeepSearch agents: with public checkpoints: Jan-Nano [4], II-Search-4B [9], Qwen3-4B [34], ZeroSearch [27], Search-o1 [15], R1-Searcher [26], WebSailor [14]. **Closed-source:** comparators: o3 [18], GPT-4o [17].

Benchmarks (9). DeepSearch (5): SimpleQA [30], FRAMES [13], WebWalkerQA [32], Seal0 [19], MuSiQue [29]. **General reasoning (4):** HLE [20], AIME-25 [1], GPQA-Diamond [21], MedQA [11]. *Metric:* Pass@1 using GPT-4.1-mini LLM as Judge (Temperature=0.).

3 Results

Fathom-Search sets a new state of the art on DeepSearch. As shown in Table 1, Fathom-Search (Stage-2) establishes a clear lead over prior open baselines. On the DeepSearch benchmark, it delivers

Table 1: **Main results.** Accuracy (%) on DeepSearch benchmarks SimpleQA, FRAMES, WebWalker, Seal0, Musique and general reasoning benchmarks HLE, AIME-25, GPQA-D, MedQA. 'Avg' is the unweighted mean within each block. Bold/italics denote best/second-best per benchmark.

	DeepSearch Benchmarks						General Reasoning Benchmarks				
Model	**SimpleQA**	**FRAMES**	**WebWalker**	**Seal0**	**Musique**	**Avg**	**HLE**	**AIME-25**	**GPQA-D**	**MedQA**	**Avg**
Closed-Source Models											
GPT-4o (without search)	34.7	52.4	3.2	7.2	34.0	26.3	2.3	*71.0*	*53.0*	*88.2*	53.6
o3 (without search)	49.4	43.2	14.0	14.0	*48.9*	33.9	*20.3*	**88.9**	*85.4*	**95.4**	*72.5*
GPT-4o (with search)	*84.4*	*63.7*	*31.6*	*15.3*	37.5	*46.5*	4.3	*71.0*	*53.0*	*88.2*	54.1
o3 (with search)	**96.0**	**86.8**	**57.0**	**49.5**	**51.2**	**68.1**	**27.4**	**88.9**	**85.4**	**95.4**	**74.3**
Open-Source Models											
Qwen-2.5-7B	3.96	16.5	2.1	1.4	6.2	6.0	1.2	10	33.0	61.2	24.7
Qwen-2.5-7B + Search	50.8	23.3	10.1	3.0	13.6	20.2	2.4	10	33.5	62.0	25.3
Qwen3-4B	3.8	14.7	2.6	2.1	9.0	6.4	4.2	*65.0*	55.1	71.0	48.8
Qwen3-4B + Search	67.7	27.2	17.5	6.2	18.7	27.5	6.2	*65.0*	*55.9*	72.0	49.8
ZeroSearch-3B	51.9	11.3	8.7	7.1	13.8	18.6	3.4	10.0	14.6	51.0	17.3
ZeroSearch-7B	75.3	30.0	18.2	6.2	20.6	30.1	4.2	10.0	29.3	57.5	22.8
R1-Searcher-7B	58.8	37.0	1.8	1.4	19.1	23.6	2.1	10.0	33.3	56.5	25.5
search-o1 (Qwen3-4B)	57.5	26.8	10.8	5.5	15.3	23.2	3.4	40.0	30.5	53.7	31.9
WebSailor-3B	87.1	44.4	**52.2**	9.0	27.4	44.0	7.4	40.0	45.5	51.3	36.0
Jan-Nano-32K	80.7	36.1	25.0	6.2	21.4	33.9	5.5	60.0	37.4	66.0	42.2
Jan-Nano-128K	83.2	43.4	33.7	6.2	23.9	38.1	6.1	53.3	51.0	65.4	44.0
II-Search-4B	*88.2*	*58.7*	40.8	17.1	*31.8*	*47.3*	*7.4*	60.0	*51.5*	*72.1*	47.8
Fathom-Search-4B (Stage-1)	88.1	57.2	39.0	*19.8*	31.3	47.1	6.7	60.0	55.6	**75.4**	*49.4*
Fathom-Search-4B (Stage-2)	**90.0**	**64.8**	*50.0*	**22.5**	**33.2**	**52.1**	**9.5**	**70.0**	**60.1**	**75.4**	**53.8**

Table 2: Ablation on using our steerable step-level reward compared to the vanilla trajectory-level RLVR reward. Trained on top of Fathom-Search-4B(Stage-1) checkpoint.

Reward	**SimpleQA**	**FRAMES**	**WebWalkerQA**	**Seal0**
Vanilla Reward (5)	88.2	58.2	43.2	21.6
Steerable Step-Level Reward (11)	90	64.8	50	22.5

52.1%, a gain of nearly 25 percentage points over Qwen3-4B + Search, and achieves 53.8% on General Reasoning with a healthy 4-point improvement. More tellingly, on the most demanding settings—such as FRAMES, WebWalkerQA, and Seal0—the model does not merely inch forward but demonstrates order-of-magnitude improvements: doubling accuracy on FRAMES and WebWalkerQA, and tripling it on Seal0. These jumps underscore that the model's advantage extends well beyond incremental progress. It also surpasses both the prior state-of-the-art system (II-Search-4B) and larger 7B-scale models like ZeroSearch-7B and R1-Searcher-7B, showing that careful reward design and training strategy can outweigh sheer parameter count.

Generalization across domains. A notable strength of Fathom-Search is its generalization to reasoning benchmarks beyond DeepSearch. Most search-augmented models degrade substantially in such settings, for instance WebSailor-3B, ZeroSearch etc show a significant degradation in benchmarks like GPQA-D compared to heir base model Qwen2.5-7B. Fathom-Search maintains strong performance on out-of-distribution benchmarks even surpassing the base model Qwen3-4B, it secures 60.1% and 75.4%, respectively—improvements of more than 20 percentage points compared to WebSailor-3B and ZeroSearch-3B. Crucially, this was achieved without any domain-specific finetuning, highlighting that the improvements are not narrow optimizations but generalizable advances in reasoning-with-retrieval.

Competing with closed-source models. The gains are not confined to open-source comparisons. Against GPT-4o + Search, closed-source baseline, Fathom-Search (Stage-2) consistently comes out ahead on SimpleQA, FRAMES, WebWalkerQA, HLE, and GPQA-D. The margins are especially pronounced on WebWalkerQA (+18.4 pp) and GPQA-D (+7.0 pp), while on HLE, Fathom-Search achieves roughly double the accuracy of GPT-4o + Search. This positions Fathom-Search as not just an open-source alternative but as a competitive option relative to premium proprietary systems.

On policy optimization: RAPO vs. GRPO. Table 3 contrasts RAPO and GRPO as the policy-optimization algorithm for Stage-1 training. When keeping the Stage-1 setup fixed, RAPO consistently outperforms GRPO across all DeepSearch benchmarks. The improvements are modest on simpler

Table 3: Ablation on using RAPO as the Policy Optimization algorithm for Stage-1 training compared to GRPO. RAPO shows superior performance on all DeepSearch tasks.

Algorithm	SimpleQA	FRAMES	WebWalkerQA	Seal0
GRPO	87.8	55.2	33.8	14.4
RAPO	88.1	57.2	39.0	19.8

datasets (e.g., +0.3 pp on SimpleQA) but grow more pronounced as task complexity increases: +2.0 on FRAMES, +5.2 on WebWalkerQA, and +5.4 on Seal0. Averaged across benchmarks, RAPO yields a +3.23 pp boost. This trend suggests that RAPO provides a more stable and effective optimization signal, particularly for multi-hop and high-variance reasoning tasks where GRPO's advantages plateau.

On the Steerable Step-Level reward. Finally, the ablation in Table 2 demonstrates the role of training signal design. Replacing the trajectory-level RLVR reward with the proposed steerable step-level reward led to steady gains across multiple benchmarks: +1.8 points on SimpleQA, +6.6 on FRAMES, +6.8 on WebWalkerQA, and a modest +0.9 on Seal0. While the improvements may appear incremental in isolation, their consistency across datasets suggests that finer-grained reward shaping not only improves raw accuracy but also stabilizes training on complex multi-step reasoning tasks.

4 Conclusion

We introduce Fathom-Search-4B and a practical post-training recipe that jointly tackles reward-sparsity, optimization instability, and shallow tool use in web-grounded reasoning. Our DUETQA multi-agent self-play corpus, two-stage RL-Zero training with RAPO, and steerable step-level rewards stabilize multi-turn learning and reliably extend tool use beyond 10 steps, fostering disciplined exploration and verification. The resulting agent attains state-of-the-art results on DeepSearch benchmarks while transferring competitively to STEM and medical evaluations, with ablations validating the gains from goal-conditioned retrieval and stronger reader LLMs. We view this as a concrete, scalable step toward dependable, autonomous DeepResearch agents.

References

[1] AIME. Aime problems and solutions, 2025, 2025.

[2] Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models, 2025.

[3] Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. Research: Learning to reason with search for llms via reinforcement learning, 2025.

[4] Alan Dao and Dinh Bach Vu. Jan-nano technical report, 2025.

[5] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting

Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[6] Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization, 2025.

[7] Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars, 2025.

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

[9] Intelligent Internet. Ii-search-4b: Information seeking and web-integrated reasoning llm. `https://huggingface.co/II-Vietnam/II-Search-4B`, 2025.

[10] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

[11] Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021.

[12] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *arXiv e-prints*, page arXiv:1705.03551, 2017.

[13] Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation, 2024.

[14] Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, Weizhou Shen, Junkai Zhang, Dingchu Zhang, Xixi Wu, Yong Jiang, Ming Yan, Pengjun Xie, Fei Huang, and Jingren Zhou. Websailor: Navigating super-human reasoning for web agent, 2025.

[15] Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. *CoRR*, abs/2501.05366, 2025.

[16] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025.

[17] OpenAI. Hello gpt-4o, 2024.

[18] OpenAI. Introducing openai o3 and o4-mini, 2025.

[19] Thinh Pham, Nguyen Nguyen, Pratibha Zunjare, Weiyuan Chen, Yu-Min Tseng, and Tu Vu. Sealqa: Raising the bar for reasoning in search-augmented language models. *arXiv preprint arXiv:2506.01062*, 2025.

[20] Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, et al. Humanity's last exam, 2025.

[21] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

[22] Serper.dev. Serper.dev – ai-powered search api.

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

[24] Kunal Singh, Sayandeep Bhowmick, Pradeep Moturi, and Siva Kishore Gollapalli. NO STRESS NO GAIN: STRESS TESTING BASED SELF-CONSISTENCY FOR OLYMPIAD PROGRAMMING. In *ICLR 2025 Workshop: VerifAI: AI Verification in the Wild*, 2025.

[25] Kunal Singh, Ankan Biswas, Sayandeep Bhowmick, Pradeep Moturi, and Siva Kishore Gollapalli. Sbsc: Step-by-step coding for improving mathematical olympiad performance, 2025.

[26] Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning, 2025.

[27] Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. Zerosearch: Incentivize the search capability of llms without searching, 2025.

[28] Shuang Sun, Huatong Song, Yuhao Wang, Ruiyang Ren, Jinhao Jiang, Junjie Zhang, Fei Bai, Jia Deng, Wayne Xin Zhao, Zheng Liu, et al. Simpledeepsearcher: Deep information seeking via web-powered reasoning trajectory synthesis. *arXiv preprint arXiv:2505.16834*, 2025.

[29] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition, 2022.

[30] Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. Measuring short-form factuality in large language models. *arXiv preprint arXiv:2411.04368*, 2024.

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

[32] Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. Webwalker: Benchmarking llms in web traversal, 2025.

[33] Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. Simpletir: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. *arXiv preprint arXiv:2509.02479*, 2025.

[34] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 technical report, 2025.

[35] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024.

[36] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.