DeepDive: Advancing Deep Search Agents with Knowledge Graphs and Multi-Turn RL

Anonymous ACL submission

Abstract

Augmenting large language models (LLMs) with browsing tools substantially improves their potential as deep search agents to solve complex, real-world tasks. Yet, open LLMs still perform poorly in such settings due to limited long-horizon reasoning capacity with browsing tools and the lack of sufficiently difficult supervised data. To address these challenges, we present DeepDive to advance deep search agents. First, we propose a strategy to automatically synthesize complex, difficult, and hard-to-find questions from open knowledge graphs. Second, we apply end-to-end multi-turn reinforcement learning (RL) to enhance LLMs' long-horizon reasoning with deep search. To encourage diversity and reduce redundancy, we design a redundancy penalty that discourages repeated similar queries. Experiments show that DeepDive-32B achieves a new open-source competitive result on BrowseComp, outperforming WebSailor, DeepSeek-R1-Browse, and Search-o1. We demonstrate that multi-turn RL training improves deep search ability and significantly contributes to the performance improvements across multiple benchmarks. We observe that DeepDive enables test-time scaling of tool calls and parallel sampling. Our code of DeepDive can be accessed at https://anonymous.4open.science/r/DeepDive-CAC6/.

1 Introduction

Large language models (LLMs)—trained with reinforcement learning (RL) using verifiable rewards—have demonstrated strong performance in complex reasoning tasks, such as mathematics and coding competitions(Wei et al., 2022; DeepSeek-AI et al., 2025; OpenAI, 2024b, 2025; Grok, 2025). As real-world tasks become increasingly complex, integrating external tools like browsing expands an LLM's knowledge beyond its training corpus. This shift requires the LLM to execute as an autonomous *agent* capable of handing complex tasks.

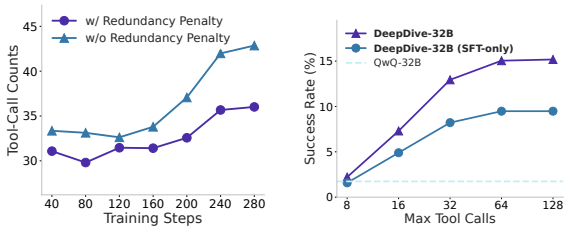


Figure 1: *Left*: Adding the redundancy penalty reduces tool call counts during RL training. *Right*: DeepDive improves deep search ability with higher maximum tool calls, boosting performance on BrowseComp.

Notably, deep search agents are expected to reason over and search from hundreds of online sources to locate complex, hard-to-find information, such as answering the questions in BrowseComp (Wei et al., 2025). However, open models fall far behind proprietary LLMs such as OpenAI DeepResearch as deep search agents (Li et al., 2025b; Song et al., 2025; Li et al., 2025c; Wu et al., 2025a). We attribute this gap to the shortage of hard-to-find data and the absence of multi-turn RL training.

First, data-wise, most existing QA datasets usually feature relatively simple questions that do not reflect true "hard-to-find" cases. For example, questions in HotpotQA (Yang et al., 2018) can often be solved by searching for a few clear entities. In contrast, deep search questions such as those in BrowseComp usually involve multiple blurry entities, requiring long-horizon reasoning and deep search to reach the correct answer. Second, training-wise, how to effectively combine long-horizon reasoning with deep search tool use remains an open question. Even strong reasoning models such as DeepSeek-R1 (DeepSeek-AI et al., 2025) make only shallow tool calls and often suffer from hallucinations. In addition, existing browsing agents that integrate browsing tools are primarily designed to address direct search tasks. Systems like R1-Searcher (Song et al., 2025), ReSearch (Chen et al., 2025), and DeepResearcher (Zheng

Table 1: Key differences between DeepDive and WebSailor (Li et al., 2025a)

	Data Generation		RL Training	
Method	**Seed Structure**	**Next-hop Constraint**	**Format Penalty**	**Search Efficiency**
WebSailor	subgraph	✗	soft	✗
DeepDive (ours)	long-chain	✓	strict	✓

et al., 2025) are mainly trained and evaluated on datasets similar to HotpotQA, including 2WikiMultiHopQA (Ho et al., 2020), Bamboogle (Press et al., 2022), and Musique (Trivedi et al., 2022).

To address these challenges, we present DeepDive to advance deep search agents. First, we automatically generate challenging QA pairs from open knowledge graphs (KGs). Second, we use end-to-end multi-turn RL to improve long-horizon reasoning in deep search scenarios.

On the data side, we address the lack of difficulty in QA datasets by automatically constructing a deep search QA dataset from KGs, as they naturally support multi-hop connections, and each entity has different attributes. By deliberately blurring some attributes of each entity during question construction, we create a form of "blurry entity". We then perform random walks on the KG to extract long, multi-hop paths and use LLMs to further obfuscate key cues, making the QA pairs more challenging. This data synthesis process produces data that effectively stimulates LLMs' long-horizon reasoning and deep search abilities.

On the training side, we adopt end-to-end multi-turn RL training to integrate reasoning with search tool use. We employ the multi-turn GRPO (Shao et al., 2024) algorithm for RL, where the LLM interacts with a web environment and receives rewards based on the final answer in the constructed QA dataset. To encourage diverse exploration and prevent redundant search behavior, we design a redundancy penalty that discourages repeated similar queries as measured by Jaccard similarity. Figure 1 (middle) shows that the RL-trained model increases tool use more effectively than baselines during inference, demonstrating test-time scaling of tool calls for improved deep search.

The DeepDive method is trained on two open models: GLM-Z1-9B-0414 (GLM et al., 2024) and QwQ-32B (Team, 2025). The constructed data consists of 3,090 high-quality deep search QAs derived from KGs. Built on this, DeepDive-32B reaches an accuracy of 15.3% on BrowseComp, surpassing many open agents—WebSailor, Search-o1, and DeepSeek-R1-Browse—and achieving a new open-source competitive result.

Table 1 summarizes our novelties over WebSailor along two axes. On the data side, DeepDive synthesizes harder deep-search questions from KGs using long-chain random walks, where the *next-hop constraint* restricts candidate transitions at each step to promote deep search and reasoning. On the training side, DeepDive enforces *strict* format validity as a hard constraint (invalid outputs receive zero reward and are terminated) and explicitly optimizes search efficiency via a redundancy-aware multi-turn RL objective. Experiments demonstrate that the performance of the DeepDive benefits significantly from the proposed end-to-end multi-turn RL training (see Figure 1 right). The main contributions are summarized as follows:

- We propose an automated method to synthesize deep search QA pairs from open KGs.
- We introduce DeepDive, an end-to-end multi-turn RL framework with a redundancy penalty that encourages diverse, efficient search.
- Built on open models, DeepDive-32B achieves 15.3% on BrowseComp and shows strong test-time scaling in tool calls and parallel sampling.

2 The DeepDive Method

We present DeepDive to advance the long-horizon information-seeking ability of deep search agents. In DeepDive, we introduce two techniques, targeting the data construction and RL stages, respectively. To generate a large-scale corpus of challenging deep-search QAs, we develop an automated and controllable data synthesis method with knowledge graphs (KG) (see Figure 2). To enhance the agent's capabilities for long-horizon reasoning and browsing, we leverage the constructed data to perform end-to-end multi-turn RL training (see Figure 3).

To mimic human web navigation, we establish an interaction framework as the learning environment for our deep search agent. The agent follows an iterative cycle of reasoning, tool execution, and

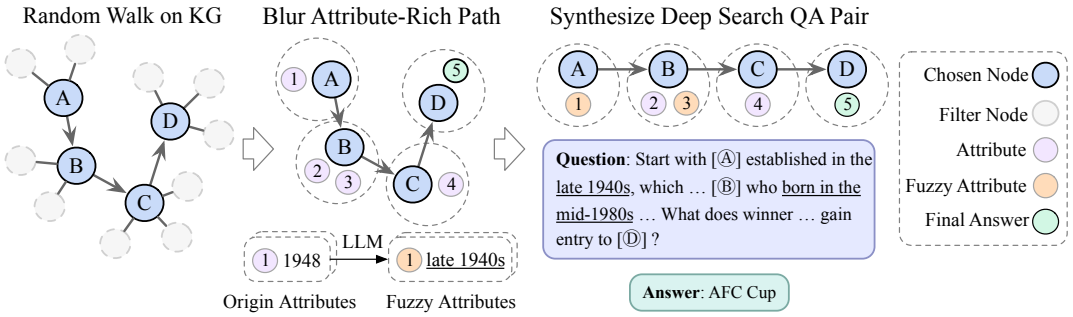


Figure 2: Overview of automated question–answer (QA) data synthesis from knowledge graphs (KGs) for DeepDive. Deep search QA pairs are automatically constructed by performing random walks over a knowledge graph and subsequently obfuscated using a large language model.

observation, followed by ReAct (Yao et al., 2023) (see Figure 3). Formally, at step t, the agent generates a chain-of-thought c_t, executes a browsing action a_t, and observes the web content o_t. This process repeats until the agent determines it has collected sufficient information and executes a terminating action a_{eos} to give the final answer. The entire task can be represented as a trajectory $\mathcal{T}$:

$$\mathcal{T} = [q, (c_1, a_1, o_1), \ldots, (c_m, a_m, o_m), c_{\text{ans}}, a_{\text{eos}}] \quad (1)$$

The action a_t is drawn from a browsing action space with three core operations: `search`, `click`, and `open`. A `search` action to retrieve web page summaries with given keywords, a `click` action to access specific pages from search results, and an `open` action to access specified URLs directly.

2.1 Automated Data Synthesis from KGs

Building deep search agents requires training data that goes beyond conventional multi-hop QA. While datasets like HotpotQA involve predictable reasoning steps, true deep search agents should act like human researchers who iteratively search, filter, and synthesize scattered evidence from the web. This thus calls for complex, difficult, and hard-to-find questions that even domain experts need hours to search and solve. However, the specific training data required to cultivate this skill is naturally scarce on the internet. With manual annotation being prohibitively expensive and difficult to scale, synthetic data generation emerges as the most efficient and scalable solution.

Knowledge Graphs with Hard-to-Find Information. Naturally, knowledge graphs (KGs) provide a structured and semantically-rich environment for multi-hop reasoning, making them particularly well-suited for generating supervision data for training deep search agents. First, *verifiability*: KGs encode factual entity-relation triples that are inherently traceable and objective, ensuring answer correctness and significantly improving data reliability compared to fully model-generated QA pairs. Second, *multi-hop structure*: KGs allow us to explicitly control reasoning depth by performing random walks of varying lengths, enabling the generation of questions requiring multiple inference steps. Third, *reasoning controllability*: each entity node contains multiple attributes that can be selectively obscured (such as dates, names, or locations), thereby increasing ambiguity and preventing models from exploiting shortcut solutions. This forces models to iteratively reason, search, and validate before finding answers. In light of these advantages, we propose an automated KG-based method to generate scalable, high-quality, and reasoning-intensive QA pairs.

Automated Data Synthesis from KGs. The main idea is to generate complex reasoning paths from KGs. A knowledge graph is a directed graph $G = (V, E)$ where V represents entities and $E \subseteq V \times V$ represents relationships between them (Ji et al., 2021). Each entity $v_i \in V$ has associated attributes $A(v) = \left[a_i^0, a_i^1, \cdots, a_i^t\right]$.

To induce deep reasoning and browsing, we sample a multi-hop reasoning path via a k-step random walk from v_0, forming $P = [v_0, \ldots, v_k]$ with valid edges (v_i, v_{i+1}). We use longer paths (e.g., $k > 5$) to increase complexity, but the raw node sequence often yields overly simple, HotpotQA-like questions that are directly searchable. To further increase the complexity of the questions, we enrich and obfuscate the path by incorporating node attributes. Specifically, we combine each node v_i in the path with its corresponding attributes to form

an attribute-rich path P_A:

$$P_A = \left[\left(v_0, \left[a_0^0, a_0^1, \ldots\right]\right), \ldots, \left(v_k, \left[a_k^0, a_k^1, \ldots\right]\right)\right] \tag{2}$$

Subsequently, we select an attribute a_k^i from the terminal node of the path, v_k, as the ground-truth answer. An LLM is then employed to obfuscate the information along the entire attribute-rich path P_A. This process involves techniques such as generalizing specific dates into ranges. The final output is a pair of challenging questions and answers (q, a_k^i), generated as follows:

$$(q, a_k^i) = \text{LLM-obscure}(P_A) \tag{3}$$

Improving Path Quality and Complexity. In a graph random walk, each step directly impacts the quality of the final path, which in turn determines the complexity and logical soundness of the generated QA pair. To improve path quality, we apply two constraints to the random walk process.

First, we filter candidate nodes by setting an appropriate out-degree range $[d_{\min}, d_{\max}]$. If a node's out-degree is excessively high, it tends to be overly popular, making answers too predictable for the model. Conversely, nodes with low out-degree may hinder effective path expansion. We thus define the next-step candidate set $\mathcal{N}(v_i)$ as:

$$\mathcal{N}(v_i) = \{u \mid (v_i, u) \in E \wedge \leq d(u) \in [d_{\min}, d_{\max}]\} \tag{4}$$

Second, to ensure logical consistency of the path, we leverage an LLM to choose the next node. Given the current path $P_i = [v_0, \ldots, v_i], i < k$, the LLM evaluates all candidates in $\mathcal{N}(v_i)$ and selects the most relevant next node to the existing path as v_{i+1}:

$$v_{i+1} = \text{LLM-select}\left(P_i, \mathcal{N}(v_i)\right) \tag{5}$$

Together, these constraints steer the random walk toward complex yet coherent reasoning paths, yielding high-quality QA pairs. To further increase difficulty, we filter questions using a frontier model (e.g., GPT-4o (OpenAI, 2024a)) with basic search: we run four attempts per question and discard any question solved in any attempt, keeping only those that fail all four runs.

2.2 End-to-End Multi-Turn RL

Given the challenging QA dataset, we use end-to-end multi-turn RL to train the agent for deep search. Based on the standard GRPO algorithm, we enhance the reward mechanism by combining strict rewards for correctness with a redundancy penalty to encourage search diversity.

Multi-Turn RL. Unlike single-turn RL, where the model outputs a single response per question, multi-turn RL lets the agent perform multiple reasoning and tool-use steps before arriving at a final answer. We employ the GRPO algorithm (Shao et al., 2024) to train the deep search agent. For each question q, we sample the tool calling trajectories G from the current policy π_θ. For each trajectory $\mathcal{T}$, we then calculate a normalized advantage $A_i = \left(r_i - \text{mean}\{r_k\}_{k=1}^G\right) / \text{std}\left\{r_k\right\}_{k=1}^G$, then the policy parameters θ are updated to maximize a clipped objective function with a KL penalty:

$$\mathcal{L}(\theta) = \frac{1}{G}\sum_{i=1}^{G}\Big[\min\Big(\rho_i A_i,\ \text{clip}\big(\rho_i, 1-\epsilon, 1+\epsilon\big)A_i\Big) - \beta\,\text{KL}(\pi_\theta \,\|\, \pi_{\text{ref}})\Big] \tag{6}$$

This objective uses the importance ratio $\rho_i = \pi_\theta(\mathcal{T}) / \pi_{\theta_{\text{old}}}(\mathcal{T})$, where ϵ controls the clipping range and β weights the penalty for diverging from a reference policy π_{ref}.

Encouraging Diverse Search with Redundancy Penalty. Deep search tasks are inherently multi-turn, as formalized in Eq. 1. Deep search tasks benefit significantly from diverse exploration strategies, as different search queries can uncover complementary information and lead to a more comprehensive understanding. To promote such diversity, we design a reward mechanism that encourages browsing agents to explore varied search approaches while maintaining correctness.

Our approach combines two key components. First, we measure search diversity by analyzing how similar queries are within a search trajectory. Given a trajectory $\mathcal{T}$ with all search queries $Q = [q_1, q_2, \ldots, q_T]$, where each query q_i contains keywords $q_i = \{w_{i,1}, w_{i,2}, \ldots, w_{i,n_i}\}$, we calculate the Jaccard similarity (Real and Vargas, 1996) between any two queries as: $\text{sim}(q_i, q_j) = |q_i \cap q_j| / |q_i \cup q_j|$. The overall similarity across all queries in the trajectory is then computed as:

$$S(\mathcal{T}) = \frac{1}{T(T-1)}\sum_{i \neq j}\text{sim}(q_i, q_j) \in [0, 1] \tag{7}$$

The metric is 1 if all queries are identical and 0 if they are completely disjoint; lower values indicate more diverse search exploration.

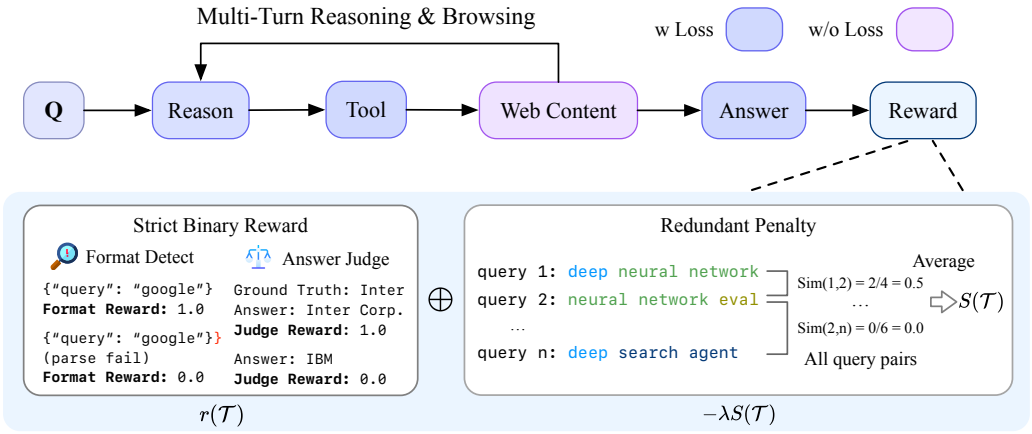


Figure 3: Overview of multi-turn RL in DeepDive.

Second, we employ a strict binary reward to ensure trajectory correctness. A trajectory $\mathcal{T}$ receives a $+1$ reward only when every step is correctly formatted, including the reason c_i and the action a_i, and the final answer a_{eos} matches the ground-truth a^*. Since entities may have multiple valid representations, we use an LLM judge (Zheng et al., 2023) for answer verification. Formally, the binary reward is defined as:

$$r(\mathcal{T}) = \begin{cases} 1, & (\forall\, i, \text{Format}\,(c_i, a_i)) \wedge \text{Judge}\,(a_{\text{eos}}, a^*) \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

We combine these components into our final reward function:

$$r'(\mathcal{T}) = r(\mathcal{T}) - \lambda \cdot S(\mathcal{T}) \tag{9}$$

where $\lambda < 1$ controls how much we reward diverse queries. This formulation encourages agents to explore a wider range of search strategies while maintaining a strong emphasis on the correctness of the final answer.

3 Experiments

3.1 Setup

Benchmarks. We evaluate DeepDive on four public and challenging deep search benchmarks: BrowseComp (Wei et al., 2025), BrowseComp-ZH (Zhou et al., 2025), Xbench-DeepSearch (Xbench-Team, 2025), and SEAL-0 (Pham et al., 2025).

Data Synthesis Details. We build synthetic datasets from two public knowledge graphs, KILT (Petroni et al., 2020) and AMiner(Tang et al., 2012). First, we generate long-chain paths via random walking with parameters set to $k \in [5, 9]$, $d = 3, d_{\min} = 4, d_{\max} = 8$. We then use Gemini-2.5-Pro (Team et al., 2023), leveraging its superior long-context ability, to obscure entities and synthesize the QA pairs. This process yields 3,250 deep search QA pairs, which are randomly split into 1,016 samples for Supervised Fine-Tuning (SFT) and 2,234 for Reinforcement Learning (RL).

Training Details. We integrate the Serper API (Serper) for web search. The Jina API (Jina.ai, 2025) handles the click and open operations. Our training process follows recent RL recipes (Guo et al., 2025; Hou et al., 2025; Li et al., 2025a), starting with a cold-start phase. We leverage the Claude-4-Sonnet-Thinking model (Anthropic, 2025a), which has tool-calling capabilities, to generate cold-start data through multiple attempts and reject sampling, yielding 858 high-quality SFT traces. We choose two open models as our backbone models: GLM-Z1-9B-0414 (GLM et al., 2024) and QwQ-32B (Team, 2025). Each model is trained for 3 epochs with a global batch size of 32, a learning rate of 1×10^{-5}, and a maximum context length of 104,800.

During RL, we conduct training using the open-source Slime framework (Zhu et al., 2025) with all 2,234 data samples. The training configuration includes a rollout size of 8, 16 samples per prompt, a global batch size of 128, a temperature of 1.0, and a maximum context length of 50,000 tokens. We set the redundancy penalty coefficient to $\lambda = 0.1$. To promote exploration, we set the KL penalty coefficient to $\beta = 0$ (Vassoyan et al., 2025) and employ a learning rate of 1×10^{-6}.

3.2 Overall Performance

Table 2 presents a comprehensive comparison between DeepDive and a range of baselines across

Table 2: Evaluation of deep search QA benchmarks. Accuracy(%) is reported. * represents reported performance from existing studies. **bold**: best among open-source models; underline: second best.

Model	Reason	Browse	BrowseComp	BrowseComp-ZH	Xbench-DeepSearch	SEAL-0
Proprietary Models						
GPT-4o	✗	✗	0.9*	11.1	18.0*	0.9
GPT-4o	✗	✓	1.9*	12.8	30.0	9.1
o1	✓	✗	9.9*	29.1*	38.0	11.7
Claude-4-Sonnet-Thinking	✓	✗	2.6	21.5	27.0	9.0
Claude-4-Sonnet-Thinking	✓	✓	14.7	30.8	53.0	37.8
DeepResearch	✓	✓	51.5*	42.9*	-	-
Open-Source Models						
GLM-Z1-9B-0414	✗	✗	0.6	2.4	8.0	7.2
GLM-Z1-9B-0414	✗	✓	0.6	1.7	3.0	2.7
QwQ-32B	✓	✗	1.7	13.5	10.7*	5.4
QwQ-32B	✓	✓	1.3	14.5	27.0	4.5
DeepSeek-R1	✓	✗	2.0	23.2	32.7*	5.4
Search-o1-32B	✓	✓	2.8*	17.9*	25.0*	-
WebDancer-32B	✓	✓	3.8*	18.0*	39.0*	-
WebSailor-7B	✓	✓	6.7*	14.2*	34.3*	-
WebSailor-32B	✓	✓	10.5*	25.5*	**53.3***	-
DeepDive-9B (*sft-only*)	✓	✓	5.6	15.7	35.0	15.2
DeepDive-9B	✓	✓	6.3	15.1	38.0	12.2
DeepDive-32B (*sft-only*)	✓	✓	9.5	23.0	48.5	23.9
DeepDive-32B	✓	✓	**15.3**	**29.7**	51.8	**25.5**

four challenging deep search benchmarks. From the results, we draw the following key observations:

Competitive among Open Search Agents. The DeepDive-32B model excels on four challenging deep search benchmarks. While most open-source models score under 10% on BrowseComp, DeepDive-32B achieved 15.3%. It also shows clear advantages on SEAL-0 and XBench-DeepSearch, indicating effective use of browsing for complex reasoning. The results also highlight the power of reinforcement learning. The 32B model with only SFT has already scored 9.5% on BrowseComp, RL then enhances the model's core ability to combine reasoning with search, resulting in stable performance growth over the SFT version on each benchmark. Notably, these gains are less pronounced for the smaller 9B model, potentially because of its limited reasoning capacity or a tendency to overfit on synthetic data during its training.

RL Drives Deeper Search Strategies. Figure 4 illustrates the effect of reinforcement learning on DeepDive-32B through two key metrics: performance and tool call counts. Evaluation accuracy on a randomly sampled subset (BrowseComp-266) consistently improves, accompanied by rising tool usage, indicating that the model explores progressively deeper search strategies. These results demonstrate that RL trained on our synthetic data successfully enhances both performance and search depth, and generalizes to unseen samples.

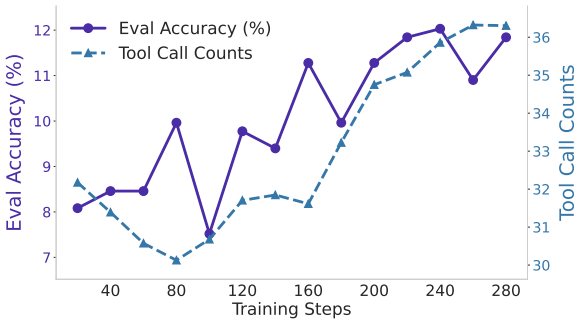
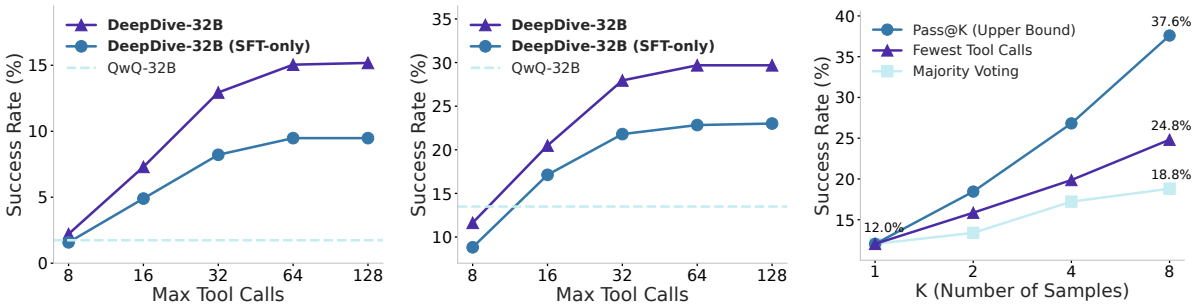


Figure 4: Evaluation accuracy and tool calls during RL training on a random subset (BrowseComp-266).

3.3 Test-Time Scaling for DeepDive

We study test-time scaling in two ways: increasing the tool-call budget for a single run, and using parallel sampling with different selection strategies. Overall, more inference compute consistently im-



(a) BrowseComp (b) BrowseComp-ZH (c) Parallel Sampling

Figure 5: Test-time scaling results for DeepDive-32B. *Left* and *Middle*: Performance vs. maximum tool calls on BrowseComp and BrowseComp-ZH (x-axis in log scale). *Right*: Parallel sampling comparison on BrowseComp-266 (a randomly sampled subset), showing that selecting answers with the fewest tool calls outperforms majority voting.

proves performance.

Tool Call Scaling during Inference. Figures 5a and 5b show that accuracy increases consistently as we raise the tool-call budget on BrowseComp and BrowseComp-ZH. When the tool call limit reaches 16 or more, DeepDive-32B trained with RL clearly outperforms its SFT-only counterpart, demonstrating the benefit of RL for tool call scaling. The dotted line indicates the QwQ-32B baseline, which is relatively low on both datasets. On BrowseComp-ZH, while QwQ-32B achieves a strong no-tool baseline (15%), DeepDive exceeds it once the budget reaches $\geq$16 tool calls.

Parallel Sampling and Tool Call Voting. Beyond scaling tool-call budgets, we study parallel sampling at test time. As shown in Figure 5c, majority voting (Wang et al., 2022) improves DeepDive-32B on BrowseComp-266 from 12.0 to 18.8. Notably, we find that samples that terminate with fewer tool calls are typically more accurate, suggesting that additional calls often reflect uncertainty rather than progress. Leveraging this, we select the answer with the fewest tool calls, which outperforms majority voting and boosts performance to 24.8, approaching the pass@8 upper bound of 37.6.

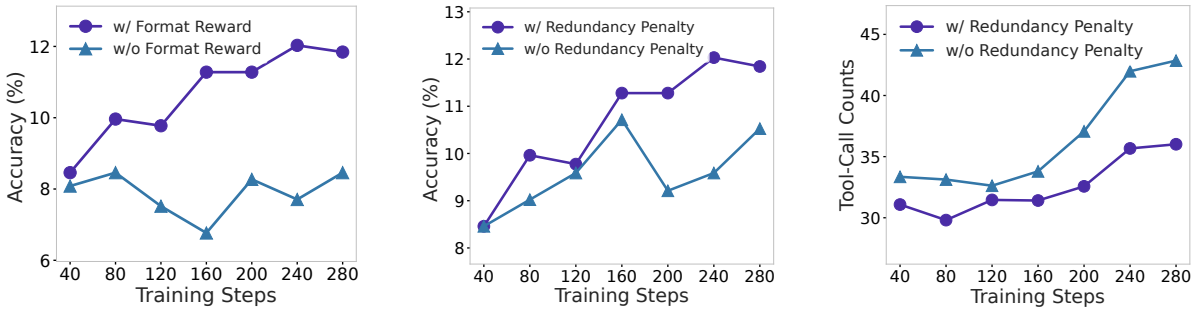
3.4 Ablation Study

Reward Ablation We evaluate two components of our reward design under identical RL settings: the strict format reward and the redundancy penalty. All models are assessed on BrowseComp-266 (a randomly sampled subset) at regular intervals (every 40 steps from 40 to 240). In Figure 6a, removing the format reward yields a curve that stays near 8.0 with almost no improvement, while adding the format reward produces a steady upward trend that remains about 2 absolute points higher throughout training. In Figure 6b and Figure 6c, adding the redundancy penalty increases accuracy in the later training phase (about 20%) and reduces tool call counts by roughly 14% under the same conditions. Overall, the strict format reward accelerates and stabilizes learning, and the redundancy penalty prunes redundant searches, improving search efficiency without sacrificing performance.

Synthetic Data Ablation We ablate synthetic deep-search QA data in both SFT and RL on the same four benchmarks as in the main experiments, reporting accuracy (Acc) and average tool calls (#Turn), where higher #Turn indicates deeper search. As shown in Table 3, the base QwQ-32B achieves low accuracy and rarely uses tools. SFT on HotpotQA trajectories provides limited gains, while SFT on our synthetic data consistently improves both accuracy and tool usage. Starting from the best SFT model, RL with HotpotQA yields only small improvements and similar tool-call patterns, whereas RL with synthetic data produces substantial gains on both metrics, most notably on BrowseComp-266. Overall, the synthetic data is critical for both stages, improving performance and enabling long-horizon search behavior.

4 Related Work

Reinforcement Learning for LLMs. Reinforcement learning has played a key role in advancing LLM alignment and reasoning. RLHF (Ouyang et al., 2022) showed that preference optimization can align models with user intent. More recent



(a) Format Reward Ablation

(b) Redundancy Penalty Ablation (Accuracy, ↑ higher is better)

(c) Redundancy Penalty Ablation (Tool counts, ↓ lower is better)

Figure 6: Ablation of our reward design. All evaluations are on a sampled subset (BrowseComp-266).

Table 3: Ablation study of different training data. For efficiency, we evaluate a subset of BrowseComp (BrowseComp-266), while the other three benchmarks are evaluated in full.

Backbone Model	**Training Data**	**BrowseComp-266**		**BrowseComp-ZH**		**XBench-DeepSearch**		**SEAL-0**	
		Acc	**#Turn**	**Acc**	**#Turn**	**Acc**	**#Turn**	**Acc**	**#Turn**
Supervised Fine-tuning (SFT)									
QwQ-32B	–	1.9	1.5	14.5	1.2	27.0	1.5	4.5	1.1
	+ HotpotQA	4.9	20.2	13.5	11.1	35.0	8.1	18.0	8.0
	+ our data	**7.5**	**32.7**	**19.0**	**24.1**	**45.5**	**15.4**	**25.2**	**13.0**
Reinforcement Learning (RL) from the best SFT model									
DeepDive-32B (*SFT only*)	+ HotpotQA	9.2	33.2	22.7	23.3	47.0	15.1	21.6	13.6
	+ our data	**12.0**	**36.3**	**29.7**	**24.9**	**50.0**	**16.7**	**25.5**	**14.5**

work emphasizes verifiable rewards for stronger reasoning: OpenAI's o1 (OpenAI, 2024b) demonstrates the effectiveness of this paradigm at scale. Meanwhile, algorithmic progress has expanded practical RL toolkits, including GRPO (Shao et al., 2024) (critic-free training), DeepSeek R1 (DeepSeek-AI et al., 2025) (strong reasoning built on GRPO), and DAPO (Yu et al., 2025) (fine-grained updates for scalable, robust pipelines).

Deep Search Agents. ReAct (Yao et al., 2023) first introduced a framework that combines reasoning and action steps, boosting LLM performance on complex tasks. Recent deep research agents, like DeepResearch (OpenAI, 2025) and Gemini Deep Research (Gemini, 2025), have reached near-expert levels in information seeking and reasoning. Open-source work spans RL-based agents that optimize tool use and retrieval (ReSearch (Chen et al., 2025), Search-o1 (Li et al., 2025b), WebThinker (Li et al., 2025c), DeepResearcher (Zheng et al., 2025), Search-R1 (Jin et al., 2025), WebShaper (Tao et al., 2025)) and framework-based systems targeting long-form deep research (OpenDeepResearch (Hugging Face, 2025), TTD-DR (Han et al., 2025)). A significant gap remains between open-source and proprietary models.

5 Conclusion

We present DeepDive that aligns deep reasoning with multi-turn web search through automated deep search QA synthesis and end-to-end multi-turn reinforcement learning. Our data pipeline generates ambiguity-rich, multi-hop questions, and our training introduces a redundancy penalty to encourage diverse and efficient search. After the RL stage, DeepDive-32B achieves 15.3% accuracy on BrowseComp, setting a new competitive standard for open-source models while surpassing larger agents and multiple strong proprietary baselines. Analyses show that complex supervision and multi-turn RL jointly ground tool use, that performance scales with tool-call budgets and parallel sampling.

Limitations

Although the two challenging deep research QA data synthesis methods we proposed enable us to generate high-quality data with guaranteed difficulty and accuracy—helping DeepDive-32B achieve competitive results among open-source models—the upper limit of difficulty remains significantly lower than datasets like BrowseComp. This limitation indirectly contributes to DeepDive-32B substantially lower performance on BrowseComp compared to advanced models such as o3 with browsing. Additionally, our training approach, which primarily targets difficult data, has led to an "over-search" phenomenon in DeepDive-32B. Determining optimal training steps and designing more appropriate reward mechanisms for the reinforcement learning stage represents an important area for future exploration.

References

Anthropic. 2025a. Building with Extended Thinking in Claude 4 Models. https://docs.anthropic.com/en/docs/build-with-claude/extended-thinking. Accessed July 16, 2025.

Anthropic. 2025b. Claude 3.7 Sonnet.

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. 2025. ReSearch: Learning to reason with search for LLMs via reinforcement learning. *arXiv preprint arXiv:2503.19470*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, and et al. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

ByteDance Doubao. 2025. Doubao.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The Llama 3 herd of models. *CoRR*, abs/2407.21783.

Gemini. 2025. Gemini deep research. https://gemini.google/overview/deep-research.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, et al. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

Grok. 2025. Grok 4. https://x.ai/news/grok-4.

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

Rujun Han, Yanfei Chen, Zoey CuiZhu, Lesly Miculicich, Guan Sun, Yuanjun Bi, Weiming Wen, Hui Wan, Chunfeng Wen, Solène Maître, et al. 2025. Deep researcher with test-time diffusion. *arXiv preprint arXiv:2507.16075*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

Zhenyu Hou, Xin Lv, Rui Lu, Jiajie Zhang, Yujiang Li, Zijun Yao, Juanzi Li, Jie Tang, and Yuxiao Dong. 2025. Advancing language model reasoning through reinforcement learning and inference scaling. *arXiv preprint arXiv:2501.11651*.

Hugging Face. 2025. Open-source deep research – freeing our search agents. Hugging Face Blog. URL: https://huggingface.co/blog/open-deep-research.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, 33(2):494–514.

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*.

Jina.ai. 2025. Jina.

Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. 2024. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation. *arXiv preprint arXiv:2409.12941*.

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. 2025a. Websailor: Navigating super-human reasoning for web agent.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025b. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*.

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

OpenAI. 2024a. Hello GPT-4o.

OpenAI. 2024b. Learning to reason with LLMs.

OpenAI. 2025. Deep research: Autonomous web-research agent. https://openai.com/index/introducing-deep-research/.

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

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 27730–27744.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, et al. 2020. Kilt: a benchmark for knowledge intensive language tasks. *arXiv preprint arXiv:2009.02252*.

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

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.

Raimundo Real and Juan M Vargas. 1996. The probabilistic basis of jaccard's index of similarity. *Systematic Biology*, 45(3):380–385.

Serper. 2025. Serper: Google search api. https://serper.dev.

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

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*.

Jie Tang, Jing Zhang, Limin Yao, Zhong Yu, Juanzi Li, Li Zhang, Zhong Su, Dong Wang, and Qiang Yang. 2012. Arnetminer: A comprehensive academic search and mining platform. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1231–1239. ACM.

Zhengwei Tao, Jialong Wu, Wenbiao Yin, Junkai Zhang, Baixuan Li, Haiyang Shen, Kuan Li, Liwen Zhang, Xinyu Wang, Yong Jiang, et al. 2025. Webshaper: Agentically data synthesizing via information-seeking formalization. *arXiv preprint arXiv:2507.15061*.

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

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Jean Vassoyan, Nathanaël Beau, and Roman Plaud. 2025. Ignore the kl penalty! boosting exploration on critical tokens to enhance rl fine-tuning. *arXiv preprint arXiv:2502.06533*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

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*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 24824–24837.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Yong Jiang, Pengjun Xie, et al. 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 Fei Huang. 2025b. Webwalker: Benchmarking llms in web traversal.

x.ai. 2025. Grok 3 beta — the age of reasoning agents.

Xbench-Team. 2025. Xbench-deepsearch.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, and et al. 2025. DeepResearcher: Scaling deep research via reinforcement learning in real-world environments. *arXiv preprint arXiv:2504.03160*.

Peilin Zhou, Bruce Leon, Xiang Ying, Can Zhang, Yifan Shao, Qichen Ye, Dading Chong, Zhiling Jin, Chenxuan Xie, Meng Cao, et al. 2025. Browsecomp-zh: Benchmarking web browsing ability of large language models in chinese. *arXiv preprint arXiv:2504.19314*.

Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. 2025. slime: An llm post-training framework for rl scaling. https://github.com/THUDM/slime. GitHub repository. Corresponding author: Xin Lv.

A Setup

Baselines. We compare DeepDive against a diverse set of models, grouped into two categories:

- **Proprietary models:** This group includes both non-browsing and browsing-capable models. Non-browsing models consist of GPT-4o (OpenAI, 2024a), Claude-3.7-Sonnet (Anthropic, 2025b), Claude-4-Sonnet-Thinking (Anthropic, 2025a) and o1 (OpenAI, 2024b), which are evaluated solely on their internal reasoning abilities. Browsing-capable proprietary models include Grok-DeepResearch (x.ai, 2025), Doubao with Deep Think and Search (Doubao, 2025), and OpenAI's Deep Research (OpenAI, 2025). Additionally, we extend select non-browsing models with our browsing tools to examine performance gains via standard function calls.

- **Open-source models:** This group includes recent high-performing open-source models, both with and without browsing capabilities. The non-browsing models consist of GLM-Z1-9B-0414 (GLM et al., 2024), DeepSeek-R1-0528 (Guo et al., 2025) and QwQ-32B (Team, 2025). We compare our method with recent open-source web agents, including Search-o1 (Li et al., 2025b), WebDancer (Wu et al., 2025a), and WebSailor (Li et al., 2025a). To ensure a fair comparison, we also enable standard function calling for GLM-Z1-9B-0414 and QwQ-32B, allowing them to browse during evaluation.

Evaluation. For datasets and models with previously-reported scores, we directly adopt the results from their respective papers. For all other evaluations, we follow the LLM-as-Judge framework (Zheng et al., 2023), employing Llama-3.1-70B (Dubey et al., 2024) to assess whether a model's final output matches the ground truth answer. To speed up evaluation during reinforcement learning (RL) training, each checkpoint is assessed on a fixed, randomly pre-sampled subset of BrowseComp-266, with a maximum of 75 turns. Once training saturates, we evaluate later checkpoints on the full BrowseComp dataset (1,266 instances) with the turn limit raised to 128. For other benchmarks whose total size is below 300, we evaluate on the entire dataset. To reduce variance and improve robustness, every dataset is evaluated twice, and the average accuracy is reported as the final result.

B Generalization on Simple Search Tasks

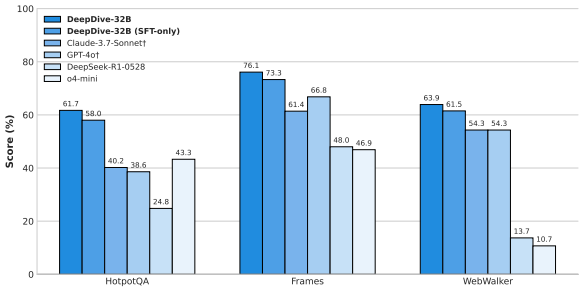


Figure 7: DeepDive-32B generalization on simple search benchmarks.

While DeepDive is trained on synthetic data based on knowledge graphs for challenging tasks like BrowseComp and BrowseComp-ZH, we evaluate its performance on simpler search benchmarks: HotpotQA (Yang et al., 2018), Frames (Krishna et al., 2024), and WebWalker (Wu et al., 2025b), which involve more direct, less ambiguous questions. We compare DeepDive against two non-search models (o4-mini and DeepSeek-R1-0528) and two proprietary search-enabled models using the same search engine. We evaluate 512 randomly selected HotpotQA questions and full test sets for other benchmarks. Figure 7 shows that both DeepDive-32B (SFT-only) and DeepDive-32B outperform all baselines, with reinforcement learning providing additional improvements across all benchmarks. These results confirm DeepDive's strong generalization and search capabilities.

C In-Depth Comparison with WebSailor

Our KG-based synthesis targets deep-search behavior by making each question more ambiguous and requiring longer evidence gathering than prior KG-derived data such as WebSailor. Although both approaches extract multi-hop paths from a knowledge graph and apply an obfuscation step, DeepDive is constructed to increase both chain length and uncertainty within a single query.

Compared to WebSailor's neighbor sampling on a Wikipedia KG, we generate paths using controlled random walks that favor long, coherent chains. We filter candidate next entities by out-degree to avoid hub nodes that create shortcuts and low-degree nodes that often terminate early. Among the remaining candidates, an LLM selects

Table 4: Difficulty comparison between WebSailor and our DeepDive data.

Method	Fuzzy Entity Number↑	R1 Accuracy↓	Open Data Size
WebSailor	3.2	40.0	20
DeepDive (ours)	**5.2**	**32.8**	**3,250**

Table 5: BrowseComp matched-budget comparison with explicit inference-time budgets. Accuracy(%) is reported. The matched-budget setting uses 32 tool calls.

Model	BrowseComp	Max Tool Counts	Context Length
Search-o1-32B	2.8	10	32k
WebDancer-32B	3.8	32	128k
WebSailor-32B	10.5	∼32	32k
DeepDive-32B	**12.9** (+23%)	32	32k
DeepDive-32B	15.3	128	128k

the next entity to keep the trajectory semantically consistent. We further restrict generation to paths longer than five hops, which better matches long-horizon search.

On top of these deeper chains, we obfuscate multiple entities per question while keeping relations and final answers unchanged, for example by fuzzing exact years or precise locations. This yields questions with several uncertain references that cannot be resolved by a single surface lookup and instead require deeper search.

We quantify this difference using all 20 released WebSailor examples and 32 randomly sampled DeepDive examples from our full set of 3,250. QwQ-32B is prompted to count fuzzy entities, and DeepSeek-R1-0528 (direct reasoning mode) is evaluated over four runs. DeepDive has a higher fuzzy-entity count (5.2 vs. 3.2) and lower R1 accuracy (32.8% vs. 40.0%), indicating higher ambiguity and difficulty. Finally, we release all 3,250 QA pairs and the full data-construction code to support reproduction and extension.

D Inference-Time Budget Control

To ensure a fair comparison, we evaluate open-source baselines under an explicitly matched tool-call limit and report the context length used in our runs in Table 5.

WebSailor-32B reports a 32k context window but does not specify a tool-call limit. Since one web page typically consumes about 1k tokens, this corresponds to an effective ceiling of roughly 32 calls. Under this matched cap, DeepDive-32B still achieves a higher BrowseComp score than WebSailor-32B (12.9 vs. 10.5). Increasing DeepDive's budget to 128 calls further raises the score to 15.3, indicating that the gains are not driven by budget asymmetry. Other open-source agents typically terminate within 3–5 calls and remain below 3.5 on BrowseComp, so they are omitted from this comparison.

E Additional Study: Semi-Automated i.i.d. Deep Search QA Synthesis for RL

We perform an additional study to directly improve model performance on deep search benchmarks. Straightforwardly, we can construct i.i.d. QA pairs with BrowseComp, whose questions are so challenging that expert annotators have to spend hours solving them, ensuring that simple search strategies are ineffective. However, reaching the depth and breadth of BrowseComp requires heavy human effort in research, annotation, and data curation. To reduce annotation costs, we present a semi-automated framework.

i.i.d. Data Synthesis. We adopt a semi-automated framework to reduce the burden on annotators, where each annotator is supported by the OpenAI o3 model (OpenAI, 2025) equipped with search capabilities and follows a four-stage process.

First, based on the nine topical domains defined in BrowseComp, the annotator collaborates with the model to identify root domains that contain abundant factual and structured web content. Second, the annotator explores various linked pages

within each root domain using the model's navigation and search features, and selects verifiable entities along with their associated attributes. Third, the annotator conducts further targeted searches related to each selected entity and engages in multi-turn interactions with the model to construct new challenging multi-hop questions. These questions are carefully written to obscure key information while retaining verifiability. Fourth, the annotator uses the model to attempt to answer the synthetic question. If the answer is incorrect or if multiple plausible answers exist, the sample is discarded. The annotator also records the time taken by the model to arrive at an answer in order to identify questions that are more difficult and of higher quality. This workflow requires minimal prior knowledge from the annotator.

Through iterative model-guided discovery, question construction, and verification, annotators can efficiently produce complex, high-quality deep search QA pairs. The same procedure is applied to Chinese websites to enhance multilingual performance. As a result, we obtain a total of 2,997 English and 275 Chinese challenging deep search QA pairs.

RL with i.i.d. Deep Search Data. We follow the same pipeline as Section 3, using SFT for cold-start and difficulty-based filtering to build a high-quality subset for RL, with all training configurations remaining identical to those before, except for the data. Table 6 presents the performance after incorporating i.i.d. training data. Notably, the DeepDive-32B-RL model achieves an accuracy of 20.8% on the full BrowseComp benchmark, representing a 40% improvement over the previous best score of 15.3% and significantly outperforming open-source alternatives. Owing to the inclusion of Chinese content in the new training corpus, the new model also demonstrates considerable gains on Chinese-language benchmarks, namely BrowseComp-ZH and Xbench-DeepSearch. Interestingly, performance on SEAL-0 remains largely unchanged, which we attribute to the dataset's focus on recognizing and selecting among different search results, which is a challenge that highlights a key area for future model enhancement.

Cost and scalability trade-off. The KG and i.i.d. pipelines have different cost profiles. The KG pipeline is fully automated once implemented, using controlled random walks followed by LLM-based obfuscation and filtering, and requires essentially no additional human annotation; in our setup it can produce on the order of 10^3 samples per day. In contrast, i.i.d. synthesis is semi-automated but human-intensive: annotators must navigate real webpages, select entities, draft questions with a strong model, and manually verify answerability and uniqueness, yielding roughly 10^2 samples per week per annotator and requiring several person-weeks to reach $\sim$3k examples. Given this trade-off, it is expected that at the same small scale, the curated i.i.d. set attains higher accuracy (22.2% vs. 15.3% on BrowseComp), while the KG pipeline is the scalable path to substantially larger corpora without additional labeling effort.

Data Contamination Analysis. To ensure that the performance improvements are not the result of data leakage, we follow the contamination analysis protocol introduced in LLaMA 2 (Touvron et al., 2023) and evaluate the i.i.d. dataset used for training. For each evaluation sample, we tokenize the input (excluding special tokens) and extract all contiguous 10-token n-grams. A token is considered contaminated if it appears in any n-gram also found in the training corpus. The contamination rate for a sample is defined as the proportion of contaminated tokens. Based on these rates, we categorize each sample into four non-exclusive subsets: Clean (less than 20% contamination), Not Clean (20% or more), Not Dirty (less than 80%), and Dirty (80% or more). As shown in Table 7, more than 97% of the samples in the dataset are classified as Clean, and there are no samples in the Dirty category. The results indicate that there is almost no test-data leakage in the constructed dataset for training.

F Case Study

Reinforcement Learning Reshapes the Model's Search Strategy Based on the sustained performance improvement on the BrowseComp-266 evaluation set during RL training, we study the model's search behavior because most of its actions involve issuing retrieval queries. Similar to human interaction with contemporary search engines like Google Search, which allow exact match quoting, logical OR aggregation and term exclusion with a leading minus sign, the retrieval interface used during training and evaluation supports these same advanced features. We therefore collected every query generated by the model when solving the evaluation set and calculated three metrics: (1) *Quote Usage*: the fraction of queries containing double quotes

Table 6: Effect of i.i.d. deep search QA data for DeepDive. DeepDive-32B Accuracy (%) on 4 deep search benchmarks with and without i.i.d. data. **bold**: best performance; underline: second best.

Model	data	BrowseComp	BrowseComp-ZH	Xbench-DeepSearch	SEAL-0
DeepDive-32B (*sft-only*)	KG data	9.5	23.0	48.5	23.9
DeepDive-32B	KG data	15.3	29.7	50.0	**25.5**
DeepDive-32B (*sft-only*)	i.i.d data	11.4	26.6	47.5	22.5
DeepDive-32B	i.i.d data	**22.2**	**33.9**	**56.0**	23.0

Table 7: Contamination analysis of BrowseComp evaluation samples using different synthetic data. Each sample is categorized based on the proportion of overlapping n-grams with the training set.

Data Type	Contamination Rate	Clean	Not Clean	Not Dirty	Dirty
KG	2.6	99.0	1.0	100.0	0.0
i.i.d.	3.4	97.7	2.3	100.0	0.0

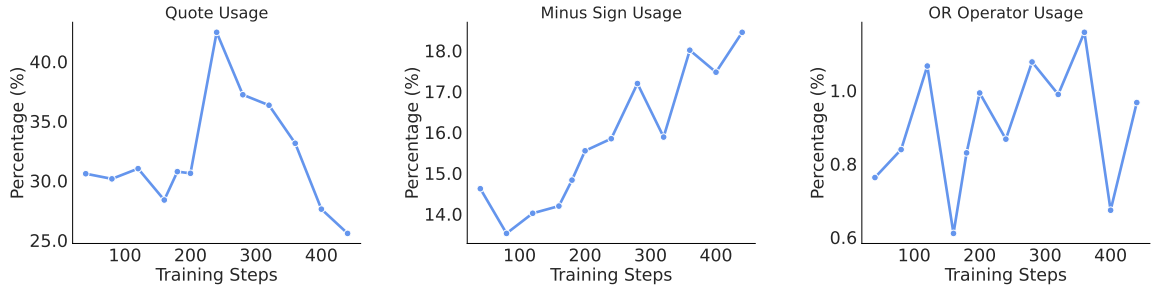


Figure 8: Evolution of *Quote Usage*, *Minus Usage* and *OR Usage* over RL training steps on BrowseComp-266.

for exact phrase matching; (2) *Minus Usage*: the fraction of queries containing a leading minus sign to exclude terms. (3) *OR Usage*: the fraction of queries containing the OR operator to combine alternative terms; The evolution of these metrics over training steps is plotted in Figure 8.

From Figure 8, we observe that Quote Usage increases from around 30% to 40% at the early stage of training, then gradually decreases to below 25%. OR Usage steadily grows from approximately 2% to 8%. In contrast, Minus Usage continues to rise from 14% to 18% throughout the training. This trend suggests that the model initially learns to adopt the quoting strategy early in reinforcement learning, but its advantage becomes less prominent over time, leading to a decline in usage. Meanwhile, the model steadily improves its ability to use minus operators, and OR Usage remains stable between 0.8% and 1%, indicating limited but consistent application.