Beyond Ten Turns: Unlocking Long-Horizon Agentic Search with Large-Scale Asynchronous RL

Jiaxuan Gao[1], **Wei Fu**[12], **Minyang Xie**[1], **Shusheng Xu**[2], **Chuyi He**[2], **Zhiyu Mei**[2], **Banghua Zhu**[3], **Yi Wu**[1*]

[1] IIIS, Tsinghua University, [2] Ant Group
[3] University of Washington
samjia2000@gmail.com, jxwuyi@gmail.com

Abstract

Recent advancements in LLM-based agents have demonstrated remarkable capabilities in handling complex, knowledge-intensive tasks by integrating external tools. Among diverse choices of tools, search tools play a pivotal role in accessing vast external knowledge. Reinforcement Learning stands out as a natural choices of learning to use tools. However, existing RL agents still fall short of achieving expert-level *Search Intelligence*, the ability to resolve ambiguous queries, analyze results, and conduct thorough exploration. Existing approaches fall short in scalability, efficiency, and data quality. For example, small turn limits in existing online RL methods, e.g. ≤ 10, restrict complex strategy learning. This paper introduces ASearcher, a large-scale RL training project of search agents. Our key contributions include: (1) *Scalable fully asynchronous RL training that enables long-horizon search* while maintaining high training efficiency. (2) *A prompt-based LLM agent that autonomously synthesizes high-quality and challenging QAs*, creating a large-scale QA dataset. Through RL training, our prompt-based 32B agent achieves substantial improvements, with +22.4 and +15.0 Avg@4 gains on xBench and GAIA, respectively. Notably, our agent exhibits extreme long-horizon search, with tool calls exceeding 100 turns and output tokens exceeding 400k during training. With a simple agent design and no external LLMs, ASearcher-Web-QwQ achieves Avg@4 scores of 51.1 on xBench and 58.1 on GAIA, achieving state-of-the-art level results.

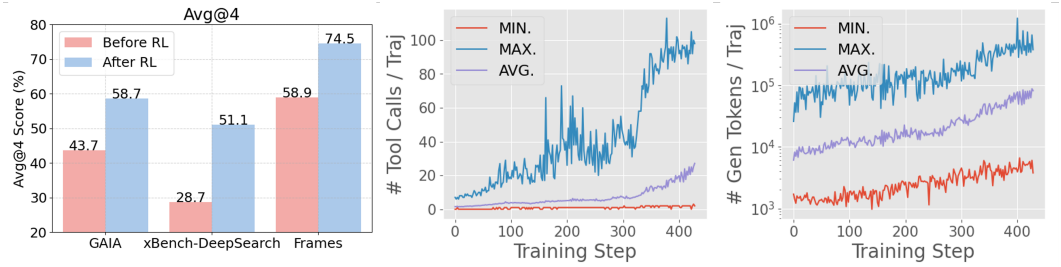

Figure 1: **(Left) Asynchronous RL brings substantial improvements:** Through RL training, our agent, ASearcher-Web-QwQ, obtains +15.0, +22.4, and +15.6 improvements on GAIA, xBench, and Frames, respectively. **(Middle) & (Right) Through RL training, ASearcher-Web-QwQ learns to conduct long-horizon search**, with tool calls exceeding 100 turns and output tokens exceeding 400k during training. The agent also **learns expert-level search strategies** (See case study in Appendix A. A)

* Corresponding author

1 Introduction

Recent advances in LLM-based agents have demonstrated remarkable capabilities in solving complex, knowledge-intensive problems by leveraging single or multiple external tools [35, 37, 31]. Among these, **search tools** stand out as particularly critical, enabling agents to access vast external knowledge for enhanced problem-solving [20, 5, 21]. However, expert-level use of search requires advanced intelligence. For instance, consider the question *"As of December 31, 2024, what were the numbers of gold, silver, and bronze medals won by China in the 2012 London Olympics?"*.While seemingly straightforward, this query is indeed challenging due to conflicting answers online (e.g., "38 gold, 27 silver, 22 bronze" vs. "39 gold, 31 silver, 22 bronze"). A search agent must navigate noisy and conflicting answers from diverse sources, identify the root cause of conflicts as doping test disqualifications from official reports, and ultimately determine the correct answer.

Online Reinforcement Learning (RL) stands out as a particularly promising direction for training models to learn advanced search strategies through trials and errors [7, 24, 27, 41]. Through online RL training, the agent could gradually learn to solve input queries with more tool calls [7, 24, 27, 41]. However, agents trained through existing online RL approaches still exhibit worse problem-solving capabilities than prompt-based LLM agents [7, 14]. In practice, we find that existing online RL approaches fail to incentivize complex and effective search strategies. We identify two critical obstacles hindering effective online RL training for search agents:

- **Insufficient search turns limit complex strategy learning.** Existing works, such as Search-R1 [7], artificially limit the number of search turns, e.g. ≤ 10 per trajectory, preventing the agent from exploring deeper search paths. However, complex queries often require multi-turn tool calls and reasoning, that could not be learned under strict turn limits.
- **Lack of large-scale, high-quality question-answer (QA) pairs:** RL training for reasoning tasks requires abundant, challenging, and correct QA pairs [1, 12, 38]. However, most existing open-source datasets for search agents are often outdated (e.g. HotpotQA), oversimplified, or too small, failing to stimulate complex search behaviors through RL [36, 13, 28].

To address these challenges, we introduce ASearcher, a *large-scale agentic RL training* project for search agents. Our contributions include:

- **Long-horizon search via fully asynchronous agentic RL training.** With a large turn limit in batch generation RL training systems [7, 24, 17, 29], long trajectories within a batch could easily lead to significant idle time, slowing down the training process. Building up on AReaL [4], our fully asynchronous system avoids long trajectories from blocking the training by decoupling trajectory execution from model updates. This allows relaxed turn limits (e.g., 128 turns/trajectory), enabling agents to explore deeper search paths without sacrificing training efficiency.
- **A scalable QA synthesis agent.** We design an LLM-based agent that autonomously generates **challenging, uncertain, and grounded** QA pairs requiring multi-turn tool use. Starting from seed questions, the agent iteratively *fuzzes queries* by obscuring key information, or *injects external facts* to increase complexity. Each constructed question undergoes *multi-stage validation* to ensure quality. We generate 25.6k high-quality samples requiring external tools to solve.

We train agents equipped with search engines and browsers under two settings, *RL training starting from base models* (Qwen2.5-7B/14B), to demonstrate that our training pipeline incentivizes strong and generalizable search strategies, and *fine-tuning a prompt-based agent empowered by a powerful LRM (QwQ-32B)*, to validate the scalability of our training pipeline in fine-tuning large-scale prompt-based LLM agents. We evaluate our agents with on multi-hop QA benchmarks and challenging benchmarks including GAIA [19] , xBench-DeepSearch [34], and Frames [10]. ASearcher-Local-7B/14B, trained only with local knowledge base, demonstrate surprisingly generalizability to realistic web search and achieve state-of-the art performances on multi-hop and single-hop QA tasks. Building up on QwQ-32B, ASearcher-Web-QwQ achieves an Avg@4 score of 51.1 on xBench-DeepSearch and 58.7 on GAIA, surpassing a set of advanced agents. Notably, through RL training, ASearcher-Web-QwQ obtains 78.0% and 34.3% improvements on xBench-DeepSearch and GAIA, respectively.

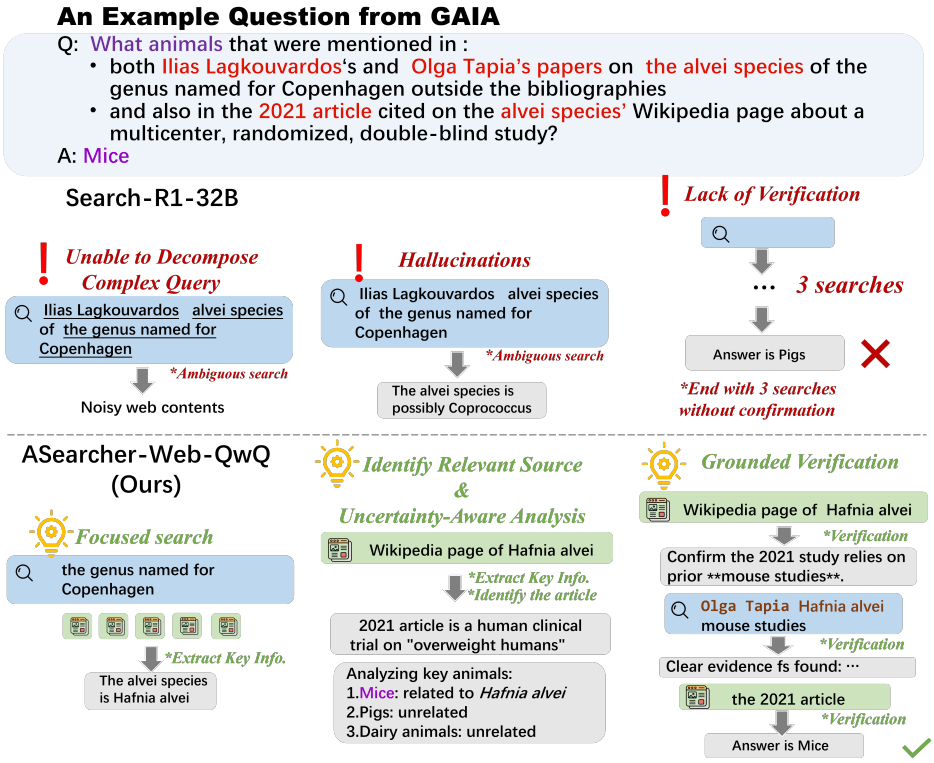

Figure 2: A case study on a complex query from GAIA. **Search-R1-32B** is unable to break down the complex question and has severe hallucinations. It is also worth noting that, since the turn limit is set as a small value, e.g. 4, during training, the model only exhibits a short tool-use horizon. Our end-to-end RL agent, **ASearcher-Web-QwQ**, exhibits key behaviors featuring Search Intelligence: *uncertainty-aware reasoning* (list and examine candidate answers), *precise extraction* from noisy contents, and *grounded verification*.

2 ASearcher

In this work, we present ASearcher, which unlocks search intelligence in search agents through large-scale asynchronous RL training. In the subsequent sections, we present the agent design, the training data as well as data synthesis agent, and fully asynchronous reinforcement learning training.

2.1 Agent Design

We employ a simple agent design in ASearcher, as illustrated in Fig. 3.

Tools. The agent can utilize two basic tools: **a search engine** and **a web browser**. The search engine takes a query as input and returns relevant snippets along with corresponding URLs. The web browser accepts a URL and returns content of the webpage. Since webpages could contain excessively long contents, therefore we employ the agent to summarize the webpage into a compact summary.

Instantiating ASearcher with Base LLMs and Advanced LRMs. We investigate two specific instantiations: either *base LLMs* such as Qwen2.5-7B/14B, or *advanced Large Reasoning Models (LRMs)* such as QwQ-32B. These two different types of instantiations require different design choices in history management and prompting.

For **base LLMs**, we following prior works [7, 24], to adopt *append-only* style prompting for the agent. Specifically, starting from a system prompt, all LLM-generated responses, search results and summaries of webpages are appended to the history.

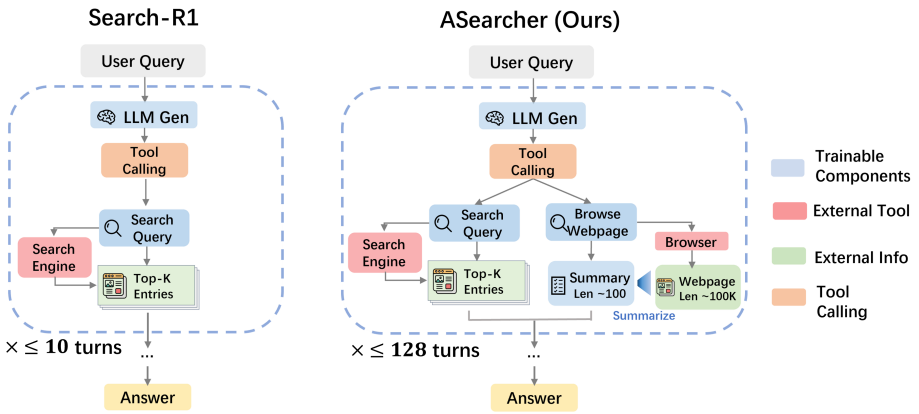

Figure 3: Comparison between ASearcher and Search-R1. (Left) Search-R1 is only equipped with search tools and lacks web browsing capability. (Right) ASearcher utilizes a simple agent design with two basic tools including search and browsing tools, without relying on any external LLM. ASearcher is a comprehensive agent capable of both reasoning and summarizing lengthy web contents.

For **LRMs**, we instruct the LRM with different prompts for tool selection, summarization, and answering. Note that LRMs typically generate long responses, and history could be long. Therefore, in the history, we discard thinking processes but instead keep summarized thoughts and tool calls. When prompting the LRM, only the most recent 25k characters of the history are provided to the LRM as additional context. These designs ensure that the LRM inputs are of at most 10k tokens.

End-to-End RL Training. Finally, we highlight that the all LLM-generated responses of the agent, including the thinking process, tool calling, and summarization, are trained using Reinforcement Learning in an end-to-end manner.

2.2 Training Data

Our training data are from two primary sources, including samples filtered from open-source datasets and synthetic high-quality question-answer (QA) pairs.

Open-source Data. We begin with the training sets from HotpotQA[36] and 2WikiMultiHopQA[6]. We employ a model-based filtering process. We first train a model on the full set of open-source data with RL, and then generate 16 responses for each question using the trained model. Finally, we filter out questions that are too hard for the model or too easy for the model. This filtering approach ensures we keep only the most challenging yet solvable questions that demand tool use. Finally, from a total of 304k QA pairs, we retain 16k challenging samples for RL training.

Data Synthesis Agent. We further develop a data synthesis agent to create high-quality question-answer pairs. As shown in Fig. 4, the data synthesis agent begins with a seed question, and iteratively modifies the question to increase the complexity. To ensure the synthetic question is strictly aligned with reliable sources, a list of *supporting facts* obtained during the question synthesis process is kept and continuously updated. At each step, the agent automatically selects between two key actions,

- **Action 1: Injection** aims to enrich the context of the question by inserting facts related to the question. The agent first selects an entity in the question and then obtains one piece of related fact about the selected entity from external sources such as Wikipedia. Then a new question is proposed by *injecting* the fact into the question.
- **Action 2: Fuzzing** blurs certain details in the question to increase the uncertainty level of the question. For example, "Catskill Mountain Railroad" could be replaced with "a historic mountain railway".

To ensure that a synthetic question is of high quality and precisely evaluate difficulty, we incorporate a rigorous *quality verification* phase. Three specific steps are included: *1. Basic Quality* by checking

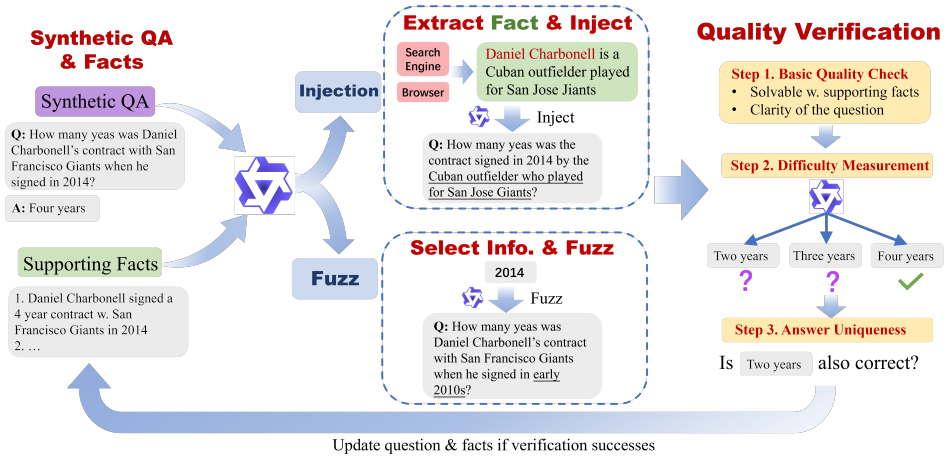

Figure 4: Data Synthesis Agent. Starting from a seed QA, the data synthesis agent iteratively modifies the question through two actions, *Injection* and *Fuzz*. Through *injection*, the agent enriches the question by adding some external facts.

the clarity of the question and verifying whether the question-answer pair is accurate based on the supporting facts; *2. Difficulty Measurement* by employing an LRM (e.g., QwQ-32B) to generate multiple answers directly without using any external tool; *3. Answer Uniqueness* by evaluating whether any of the mismatched answers generated during the Difficulty Measurement step could serve as alternative valid answers, to ensure answer uniqueness.

2.3 Asynchronous Agentic RL Training

2.3.1 Challenges of Scaling Up Trajectory Length in RL

In this section, we show that variance of trajectory execution time is large during training with a loss turn limit, which could lead to significant idle time in batch generation RL systems.

High Variance in Trajectory Execution Time. Long trajectories also introduce significant variance in execution time. We analyze the number of tool calls and token generation during RL training of our QwQ agent (Fig. 1) and observe that the longest trajectories can span dozens more tool calls and two orders of magnitude more tokens than shorter ones. This disparity leads to highly unpredictable per-trajectory runtime, further complicating training efficiency.

Efficiency Issues of Agentic RL Training. Both prolonged execution and high runtime variance degrade RL training efficiency. We take one-step-off RL training system [17] as a representative example for batch generation RL systems. As shown in Fig. 5, though this system overlaps trajectory rollouts with model training, batch generation remains bottlenecked by the slowest trajectory (e.g., trajectory 7), causing GPU idle time and under-utilization.

2.3.2 Fully Asynchronous RL Training.

To ensure efficient agentic RL training, we adopt a fully asynchronous training paradigm. Notably, our approach incorporates asynchornization at the two distinct aspects.

Asynchronous Trajectory Rollouts. Trajectory rollouts are collected in parallel and do not directly interfere with each other. Each trajectory independently sends tool calling requests to corresponding servers and LLM generation requests to the LLM inference engine. Concurrent requests from different trajectories are automatically handled by the servers. Fully independent trajectory execution ensures a trajectory does not need to wait for other trajectories when generating LLM responses and waiting for tool calling responses, thereby improving training efficiency.

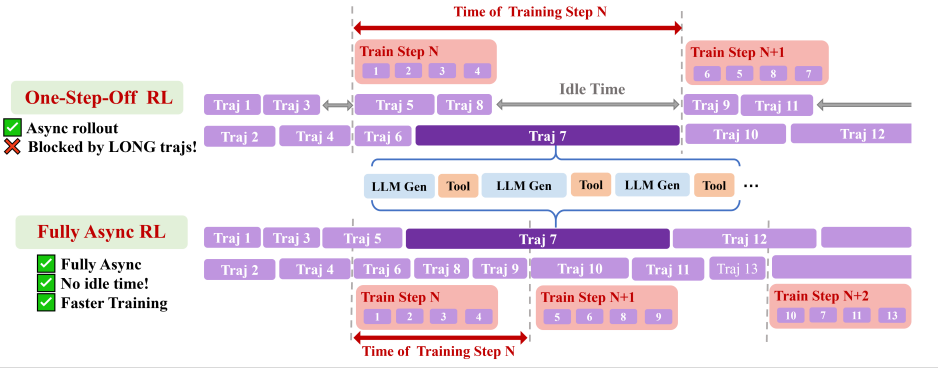

Figure 5: One-Step-off RL v.s. Fully Asynchronous RL. In batch generation systems, a batch should wait for the longest trajectory, leading to significant GPU idle time. In contrast, fully asynchronous RL achieves faster training than batch generation RL by fully decoupling training and trajectory generation, achieving near-full resource utilization for trajectory generation.

Decoupled Rollout and Training. Besides asynchronous rollout, trajectory rollouts and model updates are also fully decoupled. In Fig. 5, we compare our fully asynchronous RL training with one-step-off RL training, which utilizes asynchronous rollout within batches. In fully asynchronous RL training, long trajectories do not block generation and can span multiple versions, significantly reducing GPU idle time and achieving near-full GPU utilization during generation. On the training side, a training step is launched as soon as sufficient trajectories are collected to form a batch.

2.4 Training Details

MDP Formulation. We follow the formulation of Markov Decision Process (MDP). Formally, an MDP is defined by the tuple (S, A, T, R). Here S represents the state space, usually containing the history, search results, and retrieved webpages. A denotes the action space and an action includes tokens generated by the agent. Some tool calling could be extracted from the action through specific tags, e.g. <search> search query </search>. $T(s'|s, a)$ is the transition probability, where s' is the updated state after applying the tool calling in action a at state s. At each timestep, the agent receives a state s_t and generates an action a_t with policy $\pi : S \to A$. The goal of the agent is to maximize the return $J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} R(s_t, a_t) \middle| a_t \sim \pi(s_t)\right]$.

GRPO Training. We employ the GRPO [23] algorithm to train search agents. Specifically, for each input question x, G trajectories $\tau_1, \tau_2, \cdots, \tau_G$ are generated where $\tau_i = (s_0^i, a_0^i, s_1^i, \cdots, s_{T_i}^i)$. To optimize the agent, we employ the following loss,

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\{\tau_i\}_{i=1}^{G}\sim\pi_{\theta_{old}}(\cdot|x)} \left[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{\sum_{t=0}^{T_i-1}|a_t^i|} \sum_{t=0}^{T_i-1}\sum_{j=1}^{|a_t^i|} \min\left(\frac{\pi_\theta(a_{t,j}^i|s_t, a_{t,<j}^i)}{\pi_{\theta_{old}}(a_{t,j}^i|s_t, a_{t,<j}^i)}\hat{A}_i, \right.\right.$$
$$\left.\left. \text{clip}\left(\frac{\pi_\theta(a_{t,j}^i|s_t, a_{t,<j}^i)}{\pi_{\theta_{old}}(a_{t,j}^i|s_t, a_{t,<j}^i)}, 1-\epsilon, 1+\epsilon \right)\hat{A}_i \right) \right] \tag{1}$$

where ϵ is a hyperparameter, and $\hat{A}_i$ is the advantage for the i-th trajectory, computed based on the relative rewards of all trajectories within each group.

Dynamic Filtering. To enhance training efficiency, we implement dynamic filtering to exclude queries that lack meaningful training signals. Specifically, we remove queries where all responses yield identical rewards (resulting in zero advantages).

Reward Function. For reward function, we adopt a sparse-reward setting where rewards are computed at trajectory completion. When training from base LLMs, the reward function combines

Table 1: Results with Local Knowledge Base.

Method	Multi-Hop QA								Single-Hop QA						Avg.	
	2WikiMQA		HotpotQA		Bamboogle		Musique		NQ		TriviaQA		PopQA			
	F1	LasJ	F1	LasJ	F1	LasJ	F1	LasJ	F1	LasJ	F1	LasJ	F1	LasJ	F1	LasJ
7B Models																
Qwen-2.5-7B Direct Gen.	30.4	29.4	29.2	30.9	37.2	42.4	11.8	11.0	27.9	29.4	50.4	59.8	21.5	20.5	29.8	31.9
Search-R1-7B	54.7	58.1	57.6	60.8	**55.8**	58.4	28.2	27.1	**58.7**	49.9	68.0	78.0	57.3	55.7	54.3	55.4
R1-Searcher-7B	64.0	67.1	57.1	61.0	51.8	56.0	28.7	27.3	51.2	49.1	62.0	72.8	50.9	49.5	52.2	54.7
ASearcher-Local-7B	**72.3**	**77.6**	**62.6**	**67.6**	55.0	**60.0**	**34.4**	**32.6**	55.6	**54.5**	**68.1**	**79.3**	**57.9**	**55.9**	**58.0**	**61.0**
14B/32B Models																
QwQ-32B Direct Gen.	34.6	35.4	37.1	40.2	56.9	61.6	16.8	16.1	36.9	38.2	65.4	75.8	27.9	26.3	39.4	41.9
Search-R1-14B	48.2	49.8	56.2	58.9	52.8	51.2	27.0	25.7	60.0	51.2	71.0	79.9	56.1	54.3	53.0	53.0
Search-R1-32B	63.1	67.5	60.5	64.0	**60.0**	61.6	34.4	32.9	**60.8**	52.2	**72.0**	82.1	**60.3**	**58.2**	58.7	59.8
ASearcher-Local-14B	**72.2**	**79.1**	**65.1**	**71.0**	59.4	**64.8**	**35.6**	**34.6**	56.6	**56.1**	71.6	**84.0**	57.6	55.9	**59.7**	**63.6**

a format reward and F1 score through multiplication. When fine-tuning LRM-based agents (e.g., QwQ), we utilize LLM-as-Judge[16][32] as the reward function and omit format rewards, as these models inherently maintain proper output formatting.

3 Experiments

3.1 Experiment Setup

Benchmarks. We first evaluate the agents on single-hop and multi-hop QA tasks. For single-hop questions, we use Natural Questions [11], TriviaQA [8] and PopQA [18]. For multi-hop questions, we use HotpotQA [36], 2WikiMultiHopQA [6], MuSiQue [30], and Bamboogle [22]. We further perform evaluation on more challenging benchmarks, including Frames [10], GAIA [19], and xBench-DeepSearch [34] as extra test sets. We evaluate our approach on 1000 randomly sampled instances from the validation sets of HotpotQA, 2WikiMultiHopQA, and MuSiQue. For Bamboogle, Frames, GAIA and xBench-DeepSearch, we use their full test sets. For GAIA, we use the 103 examples from the text-only validation subset [14].

Search Tools. We evaluate the search agents with two settings, each with different types of search tools. In the first setting, **local knowledge base with RAG**, agents interact with a locally deployed RAG system to retrieve related information from a Wikipedia 2018 corpus [9]. In the other **web-based search and browsing** setting, agents operate in an interactive web environment with access to both a search engine and a browser tool.

Baselines We consider two groups of baselines aligned with the two benchmark categories. For the multi-hop and single-hop QA benchmarks, we include Search-R1(7B/14B/32B) [7], R1-Searcher(7B) [24], Search-o1(QwQ-32B) [14], DeepResearcher [41] and SimpleDeepSearcher [25]. We also prompt Qwen-2.5-7B/32B to directly generate answers without using any tools. On the more challenging benchmarks, we compare against powerful 32B-scale models, including direct generation with QwQ-32B, Search-o1(QwQ-32B) [14], Search-R1-32B [7], WebThinker-QwQ [15],SimpleDeepSearcher-QwQ [25] and WebDancer-32B [33]. All baselines are evaluated using the same tools as our agent to ensure a fair comparison.

Evaluation Metrics We adopt two evaluation metrics: F1 score and LLM-as-Judge (LasJ). For LLM-as-Judge, a strong LLM (Qwen2.5-72B-Instruct) is prompted to assess the correctness of outputs.

Training Details of ASearcher. We set the turn limit as 32 for 7B and 14B models, and 128 for ASearcher-Web-QwQ. The batch size is set as 128 for 7B and 14B models, and 64 for ASearcher-Web-QwQ.Training of ASearcher-Web-QwQ takes approximately 16k H800 GPU hours.

Table 2: Results with Web-based Search and Browsing.

Method	Training Setting	Multi-Hop QA								Single-Hop QA						Avg.	
		2WikiMQA		HotpotQA		Bamboogle		Musique		NQ		TriviaQA		PopQA			
		F1	LasJ	F1	LasJ	F1	LasJ	F1	LasJ	F1	LasJ	F1	LasJ	F1	LasJ	F1	LasJ
7B Models																	
Qwen-2.5-7B Direct Gen.	-	30.8	30.9	28.6	29.5	37.2	39.6	10.6	1.9	29.6	29.9	51.2	59.3	19.8	17.4	29.7	29.8
Search-R1-7B	local	58.9	64.8	59.0	62.8	66.3	73.6	29.4	25.4	**58.4**	51.1	73.1	84.1	**53.0**	**51.3**	56.9	59.0
R1-Searcher-7B	local	66.6	69.4	56.8	61.6	62.8	72.0	28.7	25.3	49.6	48.7	67.6	79.5	46.5	45.2	54.1	57.4
DeepResearcher-7B	web	61.0	64.1	57.1	61.0	**68.8**	**76.8**	26.8	24.5	52.0	52.9	70.0	82.8	48.9	45.7	54.9	58.3
Simple DS-7B	web	67.4	73.9	57.6	62.5	61.5	72.0	26.4	26.2	43.9	53.1	73.9	85.4	43.7	48.8	53.5	60.3
ASearcher-Local-7B	local	**69.1**	**75.5**	61.6	67.1	66.2	76.0	**33.3**	**30.7**	54.7	**53.7**	**75.2**	**87.3**	52.9	49.7	**59.0**	**62.9**
ASearcher-Web-7B	web	67.5	73.3	**61.7**	**67.2**	66.4	72.0	32.9	29.6	55.2	55.4	74	85.7	52.4	48.9	58.6	61.7
14B/32B Models																	
QwQ-32B Direct Gen.	-	33.7	33.4	39.1	42.1	56.9	57.9	18.8	19.3	37.8	43.0	63.8	74.2	25.9	24.5	39.4	42.1
Search-o1 (QwQ-32B)	-	68.9	77.8	58.4	65.3	68.6	82.4	31.8	33.5	43.1	**57.2**	**76.3**	**89.6**	43.2	48.3	55.8	64.9
Search-R1-14B	local	51.8	53.8	55.3	58.6	67.4	75.2	29.8	26.9	57.7	49.6	74.4	83.9	51.0	49.8	55.4	56.8
Search-R1-32B	local	63.7	69.3	60.3	64.2	**76.4**	**81.6**	33.0	30.8	**58.6**	51.1	76.2	86.6	**55.0**	**53.6**	60.4	62.5
Simple DS-QwQ	web	71.7	80.4	62.0	67.5	73.2	83.2	33.3	32.9	45.7	55.3	77.2	90.2	45.5	47.8	58.4	65.3
ASearcher-Local-14B	local	70.4	79.8	**63.6**	70.5	68.7	80.8	35.1	33.8	53.5	55.4	76.1	88.5	52.5	50.5	60.0	**65.6**
ASearcher-Web-14B	web	**76.1**	**80.7**	63.5	**68.5**	69.9	75.2	**36.6**	**33.7**	56.0	55.5	75.4	87.6	52.9	50.0	**61.5**	64.5

Table 3: Results on GAIA, xBench-DeepSearch, and Frames. The results are evaluated with LLM-as-Judge. For baselines, we run for 4 seeds and report Avg@4 and Pass@4.

Method	GAIA		xBench-DeepSearch		Frames	
	Avg@4	Pass@4	Avg@4	Pass@4	Avg@4	Pass@4
QwQ-32B Direct Gen.	23.1	31.1	11.8	23.0	29.9	39.9
Search-o1 (QwQ)	48.1	67.0	40.3	65.0	63.6	81.1
Search-R1-32B	28.6	43.7	19.5	37.0	44.1	61.0
WebThinker-QwQ	42.5	57.3	32.8	52.0	57.7	79.5
Simple DS-QwQ	47.6	64.1	35.8	61.0	67.0	82.2
WebDancer-QwQ	47.4	61.2	40.0	68.0	63.8	81.4
ASearcher-Web-QwQ	**58.7**	**74.7**	**51.1**	**75.0**	**74.5**	**85.5**

3.2 Main Results

Local Knowledge Base with RAG on Standard QA Benchmarks. As shown in Table 1, ASearcher-Local, trained via reinforcement learning with local knowledge base, achieves the best performance across 7B and 14B on a suite of multi-hop and single-hop QA benchmarks. In the 7B setting, ASearcher attains an average F1 of **58.0**, outperforming strong baselines such as Search-R1-7B (54.3) and R1-Searcher-7B (52.2). It also achieves a LasJ score of **61.0**, significantly outperforming Search-R1-7B (55.4) and R1-Searcher-7B (54.7). The gains are even more pronounced at the 14B scale, where ASearcher-Local-14B reaches an F1 of **60.0** and LasJ of **65.6**, surpassing even the larger 32B retrieval-based baseline Search-R1-32B.

Web-based Search and Browsing on Standard QA Benchmarks In Table 2, we evaluate agents with realistic search engines. Across both model sizes, ASearcher consistently outperforms strong baselines. In particular, ASearcher-Web-14B achieves the best performance, surpassing SimpleDeepSearcher, the strongest 32B baseline in this setting. Remarkably, ASearcher-Local-14B model exhibits strong generalization when tested in the web-based setting, achieving significant gains over all baselines in terms of LasJ. This confirms that ASearcher learns generalizable search strategies that transfer to different information sources.

Web-based Search and Browsing on Challenging Benchmarks. Table 3 shows experiment results on challenging QA tasks that require advanced problem-solving capabilities and search strategies. As a result, directly generating answers from models (e.g., QwQ-32B) performs poorly across all datasets. Our agent, ASearcher-Web-QwQ, achieves the best Avg@4 scores on GAIA (58.7) and xBench-DeepSearch (51.1), outperforming previous state-of-the-art open-source agents. These results further highlight superiority in handling long-horizon planning, real-world tool use,

and open-domain exploration. Besides Avg@4, we also report the Pass@4 score that computes the ratio of questions that an agent finds the correct answer out of 4 trials. ASearcher-Web-QwQ also outperforms state-of-the-art open-source agents in terms of pass rate.

Effect of RL Training. As shown in Fig. 1, ASearcher-Web-QwQ obtains +15.0, +22.4, and +15.6 improvements on GAIA, xBench-DeepSearch and Frames respectively. When considering the pass rate, i.e. Pass@4, ASearcher-Web-QwQ also obtains significant gains, especially on xBench-DeepSearch with 24.0 improvements. A detailed case study showing that ASearcher-Web-QwQ learns advanced search strategies is given in Appendix A.

4 Related Works

Search Agents. Some works have constructed agent workflows that enable LLMs to leverage tools for solving complex tasks, including Search-o1[14] and ReAgent[40]. Prompt-based methods, though effective for rapid development, are fundamentally limited by the capacity of the underlying LLMs. Some works attempt to construct SFT trajectories for LLMs. For instance, [2, 39] leverage large LLMs to synthesize retrieval and reasoning trajectories to fine-tune smaller models. Recently, some works investigate Reinforcement learning (RL) methods to enhance the LLM-based agents, mostly focusing on multi-hop QA benchmarks. [7, 24, 3, 41] perform RL training with multi-hop QA data and observe an increasing amount of tool calls. RAG-R1 [26] further combines SFT and RL to enhance the search strategies. More recently, researchers have begun to focus on more challenging tasks, by fine-tuning complex prompt-based agents powered by LRMs through offline RL [15], SFT on simulated real-world web data [25, 13], and constructing challenging QAs for RL training. [28].

Synthetic Data for Search Agents. Rather than relying on large-scale human annotation, data synthesis has emerged as a scalable approach to prepare training data for search agents. Recent approaches generate realistic QA trajectories by interacting with real web pages and curating data using LRMs [25, 33, 13]. On the other hand, WebSailor [13] constructs structurally challenging tasks through sampling and fuzzing, and WebShaper [28] utilizes techniques from set theory to construct high-quality complex QAs. By contrast, ASearcher develops an autonomous LLM agent for synthesizing challenging QAs with high uncertainty, without relying on complex knowledge graphs.

5 Conclusion

In this work, we present ASearcher, investigating large-scale RL training for search agents. Our contribution includes a fully asynchronous agentic RL training system and a data synthesis agent for large-scale high-quality QA construction. By instantiating ASearcher with base LLMs including Qwen2.5-7B/14B and prompt-based LLM agents based on QWQ-32B, ASearcher outperforms state-of-the-art agents across different model sizes and evaluation settings. With fully asynchronous agentic RL training and insight from our data synthesis pipeline, we hope our work could benefit future work on training advanced agents for a broader range of applications.

References

[1] 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. URL `https://hkunlp.github.io/blog/2025/Polaris`.

[2] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Self-reflective retrieval augmented generation. In *NeurIPS 2023 workshop on instruction tuning and instruction following*, 2023.

[3] Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, Fan Yang, et al. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*, 2025.

[4] Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, Tongkai Yang, Binhang Yuan, and Yi Wu. Areal: A large-scale

asynchronous reinforcement learning system for language reasoning, 2025. URL https://arxiv.org/abs/2505.24298.

[5] Google Team. Introducing Gemini deep research, 2025. URL https://gemini.google/overview/deep-research/. Accessed: 2025-04-06.

[6] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps, 2020. URL https://arxiv.org/abs/2011.01060.

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

[8] Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

[9] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781, 2020.

[10] 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. URL https://arxiv.org/abs/2409.12941.

[11] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://doi.org/10.1162/tacl_a_00276.

[12] Jiazheng Li, Hong Lu, Kaiyue Wen, Zaiwen Yang, Jiaxuan Gao, Hongzhou Lin, Yi Wu, and Jingzhao Zhang. Questa: Expanding reasoning capacity in llms via question augmentation. *arXiv preprint arXiv:2507.13266*, 2025.

[13] Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, et al. Websailor: Navigating super-human reasoning for web agent. *arXiv preprint arXiv:2507.02592*, 2025.

[14] 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. *arXiv preprint arXiv:2501.05366*, 2025.

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

[16] Yuxuan Liu, Tianchi Yang, Shaohan Huang, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. Calibrating llm-based evaluator. *arXiv preprint arXiv:2309.13308*, 2023.

[17] Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepcoder: A fully open-source 14b coder at o3-mini level. https://pretty-radio-b75.notion.site/DeepCoder-A-Fully-Open-Source-14B-Coder-at-O3-mini-Level-1cf81902c14680b3bee5eb349a512a51, 2025. Notion Blog.

[18] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khashabi. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. *arXiv preprint*, 2022.

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

[20] OpenAI. Introducing deep research, 2025. URL https://openai.com/index/introducing-deep-research/. Accessed: 2025-04-06.

[21] Perplexity Team. Introducing Perplexity deep research, 2025. URL https://www.perplexity.ai/hub/blog/introducing-perplexity-deep-research. Accessed: 2025-04-06.

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

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

[24] 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. *arXiv preprint arXiv:2503.05592*, 2025.

[25] Shuang Sun*, Huatong Song*, Yuhao Wang, Ruiyang Ren, Jinhao Jiang, Junjie Zhang, Lei Fang, Zhongyuan Wang, and Ji-Rong Wen Wayne Xin Zhao. Simpledeepsearcher: Deep information seeking via web-powered reasoning trajectory synthesis. 2025. URL https://github.com/RUCAIBox/SimpleDeepSearcher.

[26] Zhiwen Tan, Jiaming Huang, Qintong Wu, Hongxuan Zhang, Chenyi Zhuang, and Jinjie Gu. Rag-r1: Incentivize the search and reasoning capabilities of llms through multi-query parallelism. *arXiv preprint arXiv:2507.02962*, 2025.

[27] Zhiwen Tan, Jiaming Huang, Qintong Wu, Hongxuan Zhang, Chenyi Zhuang, and Jinjie Gu. Rag-r1 : Incentivize the search and reasoning capabilities of llms through multi-query parallelism, 2025. URL https://arxiv.org/abs/2507.02962.

[28] Zhengwei Tao, Jialong Wu, Wenbiao Yin, Junkai Zhang, Baixuan Li, Haiyang Shen, Kuan Li, Liwen Zhang, Xinyu Wang, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Webshaper: Agentically data synthesizing via information-seeking formalization, 2025. URL https://arxiv.org/abs/2507.15061.

[29] Prime Intellect Team, Sami Jaghouar, Justus Mattern, Jack Min Ong, Jannik Straube, Manveer Basra, Aaron Pazdera, Kushal Thaman, Matthew Di Ferrante, Felix Gabriel, et al. Intellect-2: A reasoning model trained through globally decentralized reinforcement learning. *arXiv preprint arXiv:2505.07291*, 2025.

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

[31] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

[32] Minzheng Wang, Longze Chen, Cheng Fu, Shengyi Liao, Xinghua Zhang, Bingli Wu, Haiyang Yu, Nan Xu, Lei Zhang, Run Luo, et al. Leave no document behind: Benchmarking long-context llms with extended multi-doc qa. *arXiv preprint arXiv:2406.17419*, 2024.

[33] Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Gang Fu, Yong Jiang, et al. Webdancer: Towards autonomous information seeking agency. *arXiv preprint arXiv:2505.22648*, 2025.

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

[35] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101, 2025.

[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. *arXiv preprint arXiv:1809.09600*, 2018.

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

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

[39] Tian Yu, Shaolei Zhang, and Yang Feng. Auto-rag: Autonomous retrieval-augmented generation for large language models. *arXiv preprint arXiv:2411.19443*, 2024.

[40] Xinjie Zhao, Fan Gao, Xingyu Song, Yingjian Chen, Rui Yang, Yanran Fu, Yuyang Wang, Yusuke Iwasawa, Yutaka Matsuo, and Irene Li. Reagent: Reversible multi-agent reasoning for knowledge-enhanced multi-hop qa. *arXiv preprint arXiv:2503.06951*, 2025.

[41] Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *arXiv preprint arXiv:2504.03160*, 2025.

A Full Case Study

In this section, we provide a detailed case study on an extremely challenging question from GAIA [19]. Specifically, we analyze Search-R1-32B [7] and Search-o1 (QwQ) [14] in Fig. 6.

Solution Path of the Sample Question. In Fig. 6, our case study is carried out on a question requiring finding some specific animal given 2 **conditions** and 4 **unknown variables**. To identify the correct answer, the search agent should first find out the mentioned species **U1** according to condition **C1**, identify the correct article **U2** that satisfies condition **C2**, and then find out the papers listed in **U3.1** and **U3.2**. Finally, the correct answer should be determined by cross referencing the article **U2** and the papers **U3.1**&**U3.2**. To summarize, this example is challenging for several main reasons,

- **High Uncertainty:** The question involves multiple unknown variables that could point to many different entities. For example, the 2021 article **U2** could point to any article published in 2021 and could only be determined given the condition **C2** and the alvei species **U1**.
- **Requirement for Exact Information Extraction:** To find the answer, the agent should list all animals mentioned on the webpages and making cross-document comparison. This would require the agent to precisely extract key information from the vast, noisy web contents, instead of simply summarizing the webpages.
- **Misleading Answers:** During the process of solving this task, there could be multiple misleading answers, such as "pigs". The agent should rigorously confirm its conclusions by checking the intended answer in all related webpages and documents.

Existing Online RL Approaches Fail to Learn Complex Search Strategies. In Fig. 6, Search-R1-32B is not able to decompose the complex query into individual components, consequently only making redundant queries that involve too many unknown information. The agent also has severe hallucinations, producing conclusions that are not supported by the search results. Finally, it fails to resolve all unknown variables. This case study shows that existing online RL approaches only incentivize elementary search strategies. It is also worth noting that, since the turn limit is set as a small value, e.g. 4, during training, the model only exhibits a short tool-use horizon.

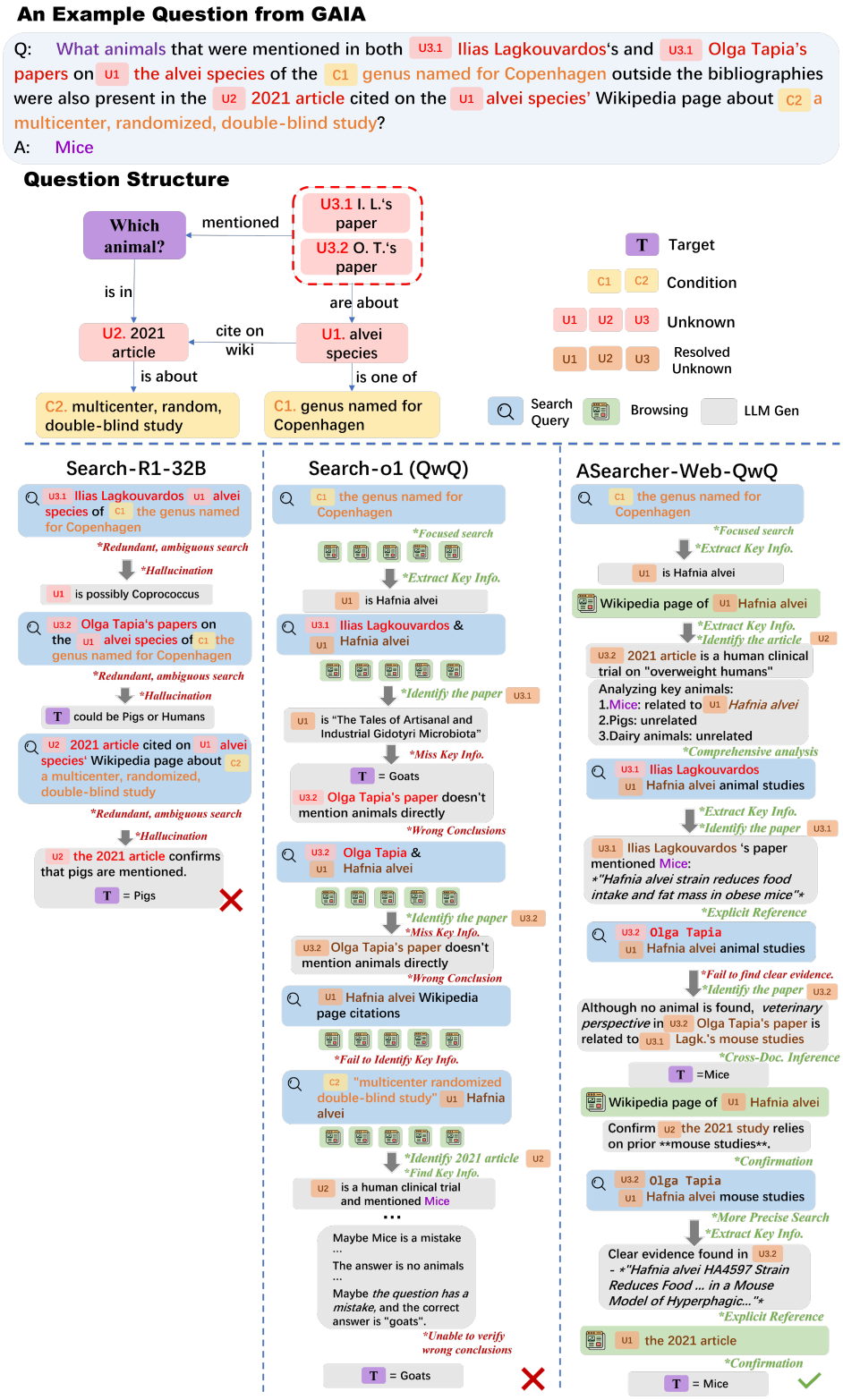

Figure 6: A case study on a complex query from GAIA. **Search-R1-32B** is unable to break down the complex question and has severe hallucinations. **Search-o1 (QwQ)** can identify the corrects articles through extensive tool calls, but easily misses key information and fails to verify wrong conclusions. Our end-to-end RL agent, **ASearcher-Web-QwQ**, exhibits key behaviors featuring Search Intelligence: *uncertainty-aware reasoning* (list and examine candidate answers), *precise extraction* from noisy contents, *cross-document inference*, and *rigorous confirmation*.

Prompt-based LLM Agents Could Fail Due to Insufficient Capability of the LLM. In Fig. 6, Search-o1 (QwQ) can find the species name **U1**, as well as the 2021 article **U2** and papers **U3.1&U3.2** through a large amount of tool calls. However, when trying to find the answer, Search-o1 (QwQ) would easily miss key information. Consequently, the agent makes incorrect conclusions. Notably, even when the agent finds information that directly links to the correct answer, it is still misguided by previous incorrect conclusions. Finally, the agent is unable to verify the correctness of previous conclusions. This case study reveals that, though an open-source model that is not explicitly trained on agentic tasks can perform extensive tool calls, it could not make expert-level reasoning based on the retrieved contents and history contexts.

ASearcher-Web-QwQ. We also analyze the search strategy of our end-to-end RL agent, ASearcher-Web-QwQ.As shown in Fig. 6, ASearcher-Web-QwQ decomposes the complex query into precise and focused queries. Unlike Search-o1 (QwQ) that visits a large amount of websites after each search query, ASearcher-Web-QwQ focuses on visiting the most relevant website. ASearcher-Web-QwQ summarizes all related information from a website. Specifically, all candidate answers are listed and carefully analyzed by the agent. When trying to search for related facts in the papers **U3.1&U3.2**, the agent explicitly references the key information. When the search results do not directly point to the desired target, e.g. when searching with "Olga Tapia (**U3.2**) Hafnia alvei (**U1**) animal studies" to find the animals related to Olga Tapia's paper, the agent does not get a clear information but is able to infer the correct answer by make connection with the other paper **U3.1**. After the correct answer "Mice" is found, the agent spends further turns on confirming previous conclusions before reporting the final answer. In summary, ASearcher successfully train a search agent that exhibits complex behaviors that feature Search Intelligence,

- **Uncertainty-aware reasoning:** the agent exhaustively lists and examines all possibilities for uncertain entities
- **Price Key Information Extraction:** the agent is able to identify the key information from vast, noisy web contents.
- **Cross-document Inference:** the agent is able to infer critical conclusions by making connections among multiple documents.
- **Rigorous Confirmation:** the agent verifies the correctness of previous conclusions with additional tool calls.

B Data Synthesis Agent

B.1 Example of Synthetic QA

We provide two illustrative examples in Tab. 4. Starting with a simple question, the injection action replaces specific entities with related factual details. For instance, "Michael P. Hein" is expanded to "who served as the first County Executive of Ulster County, New York...". The fuzzing action introduces ambiguity by generalizing precise information, replacing the exact year "1934" with "the early 1930s" or substituting "Catskill Mountain Railroad" with "a historic mountain railway."

B.2 Syntehtic QA Statistics

Table 4: Examples of the synthetic questions, where red indicates injected facts and cyan represents fuzzed content.

Round	Action	Question
Seed QA	-	When was Michael P. Hein born?
Round 1	Injection	When was the Eckerd College alumnus who served as the first County Executive of Ulster County, New York, and graduated with a Bachelor of Arts in Business Administration born?
Round 2	Injection	When was the individual born who, as County Executive of Ulster County, New York, permitted the Catskill Mountain Railroad to continue operations between Kingston and Hurley during the 2016 United States House of Representatives elections and also held that position during the 2018 elections?
Round 3	Fuzzing	When was the individual born who, as County Executive of Ulster County, New York, permitted a historic mountain railway to continue operations between Kingston and Hurley during the 2016 United States House of Representatives elections and also held that position during the 2018 elections?
...	...	...
Seed QA	-	Where is the Riggs-Hamilton American Legion Post No. 20 located?
Round 1	Injection	Where is the American Legion Post in Russellville, Arkansas, built in 1934 and recognized as a notable example of WPA Rustic architecture and listed on the National Register of Historic Places located?
Round 2	Fuzzing	Where is the American Legion Post in Russellville, Arkansas, built in the early 1930s and recognized as a notable example of New Deal-era public works architecture and listed on the National Register of Historic Places located?
Round 3	Fuzzing	Where is the veterans' organization's building in Russellville, Arkansas, built in the early 1930s and recognized as a notable example of New Deal-era public works architecture and listed on the National Register of Historic Places located?
...	...	...

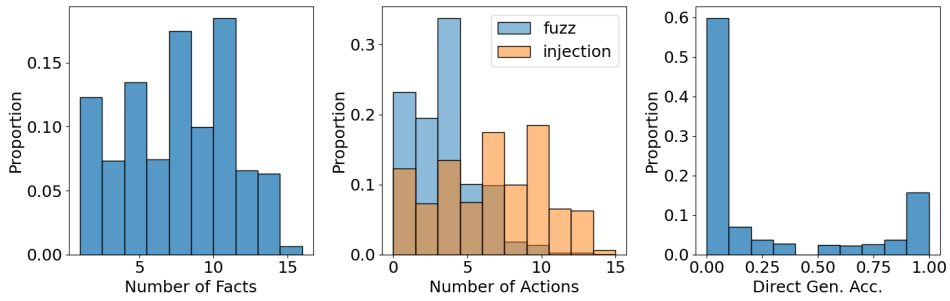

Figure 7: Statistics from our data synthesis process. (Left) The distribution of the number of supporting facts. (Middle) The distribution of the number of fuzz actions and injection actions. (Right) The accuracy distribution of QwQ-32B in answering the generated questions without using any tools.