WebDancer: Towards Autonomous Information Seeking Agency

Jialong Wu[*], Baixuan Li[*], Runnan Fang[*], Wenbiao Yin[✉][*], Liwen Zhang, Zhengwei Tao,
Dingchu Zhang, Zekun Xi, Gang Fu, Yong Jiang[✉], Pengjun Xie, Fei Huang, Jingren Zhou

Tongyi Lab, Alibaba Group

https://github.com/Alibaba-NLP/WebAgent

Abstract

Addressing intricate real-world problems necessitates in-depth information seeking and multi-step reasoning. Recent progress in agentic systems, exemplified by *Deep Research*, underscores the potential for autonomous multi-step research. In this work, we present a cohesive paradigm for building end-to-end agentic information seeking agents from a data-centric and training-stage perspective. Our approach consists of four key stages: (1) browsing data construction, (2) trajectories sampling, (3) supervised fine-tuning for effective cold start, and (4) reinforcement learning for enhanced generalisation. We instantiate this framework in a web agent based on the `ReAct`, **WebDancer**. Empirical evaluations on the challenging information seeking benchmarks, GAIA and WebWalkerQA, demonstrate the strong performance of WebDancer, achieving considerable results and highlighting the efficacy of our training paradigm. Further analysis of agent training provides valuable insights and actionable, systematic pathways for developing more capable agentic models.

1 Introduction

Web agents are autonomous systems that perceive their real-world web environment, make decisions, and take actions to accomplish specific and human-like tasks. Recent systems, such as ChatGPT *Deep Research* OpenAI (2025a) and Grok *DeepSearch* x.ai (2025), have demonstrated strong deep information-seeking capabilities through end-to-end reinforcement learning (RL) training.

The community's previous approaches for information seeking by agentic systems can be categorized into two types: *(i)* Directly leveraging prompting engineering techniques to guide Large Language Models (LLMs) or Large Reasoning Models (LRMs) Wu et al. (2025); Team (2025b); Li et al. (2025a) to execute complex tasks. *(ii)* Incorporating *search* or *browser* capabilities into the web agents through supervised fine-tuning (SFT) or RL Chen et al. (2025); Li et al. (2025a); Song et al. (2025); Jin et al. (2025); Sun et al. (2025); Zheng et al. (2025). The first training-free methods are unable to effectively leverage the reasoning capabilities enabled by the reasoning model. Although the latter methods internalize certain information-seeking capabilities through SFT or RL training, both the training and evaluation datasets are relatively simple and do not capture the real-world challenges, for instance, performance on the 2Wiki dataset has already reached over 80%. Moreover, the current SFT or RL training paradigm does not fully and efficiently exploit the potential of information-seeking behavior. Building autonomous information seeking agency involves addressing a set of challenges that span web environment perception and decision-making: (1) acquiring high-quality, fine-grained browsing data that reflects diverse user intents and rich interaction contexts, (2) constructing reliable trajectories that support long-horizon reasoning and task decomposition, and (3) designing scalable and generalizable training strategies capable of

---

[*]Equal Core Contributors. Jialong Wu and Wenbiao Yin are project leaders.
[✉]Corresponding author.

endowing the web agent with robust behavior across out-of-distribution web environments, complex interaction patterns, and long-term objectives.

To address these challenges, our objective is to **unlock the autonomous multi-turn information-seeking agency, exploring how to build a web agent like *Deep Research* from scratch.** An agent model like *Deep Research* produces sequences of interleaved reasoning and action steps, where each action invokes a tool to interact with the external environment **autonomously**. Observations from these interactions guide subsequent reasoning and actions until the task is completed. This process is optimized through end-to-end tool-augmented training. The `ReAct` framework Yao et al. (2023) is the most suitable paradigm, as it tightly couples reasoning with action to facilitate effective learning and generalization in interactive settings.

We aim to provide the research community with a systematic guideline for building such agents from a *data-centric* and *training-stage* perspective.

From a *data-centric* perspective, constructing web QA data is crucial to building web agents, regardless of whether the training paradigm is SFT or RL. Widely used QA datasets are often shallow, typically consisting of problems that can be solved with a single or a few-turn search. Previous works often filter the difficult QA pairs from open-sourced human-labeled datasets using prompting techniques Song et al. (2025). Additionally, challenging web-based QA datasets typically only have test or validation sets, and their data size is relatively small. For example, GAIA Mialon et al. (2023) only has 466, WebWalkerQA Wu et al. (2025) contains 680 examples, and BrowseComp Wei et al. has 1,266, making them insufficient for effective training. Therefore, the automatic synthesis of high-quality datasets becomes crucial. Fang et al. (2025); Zuo et al. (2025). We synthesise the datasets in two ways: 1). By crawling web pages to construct deep queries, referred to as CRAWLQA, enabling the acquisition of web information through click actions. 2). By enhancing *easy-to-hard* QA pairs synthesis to incentivize the progression from *weak-to-strong* agency, transforming simple questions into complex ones, termed E2HQA.

From a *training-stage* perspective, prior work has explored SFT or off-policy RL, but these approaches often face generalization issues, particularly in complex, real-world search environments. Other methods adopt on-policy RL directly Chen et al. (2025), but in multi-tool settings, early training steps tend to focus primarily on learning tool usage via instruction following. To enable more efficient and effective training, we adopt a two-stage approach combining rejection sampling fine-tuning (RFT) with subsequent on-policy RL. For the trajectory sampling, we restrict the action space to two commonly effective web information-seeking tools as *action*: *search* 🔍 and `click` 🖱️ . Building on this setup, we employ rejection sampling to generate trajectories using two prompting strategies: one with a strong instruction LLMs for Short-CoT and another leveraging the LRMs for Long-CoT. These yield high-quality trajectories containing either short or long `thought`, respectively. In the RL stage, we adopt the Decoupled Clip and Dynamic Sampling Policy Optimization (DAPO) algorithm Yu et al. (2025), whose *dynamic sampling* mechanism can effectively exploit QA pairs that remain underutilized during the SFT phase, thereby enhancing data efficiency and policy robustness.

Our key contributions can be summarized as follows: we abstract the end-to-end web agents building pipeline into four key stages: **Step** I: Construct diverse and challenging deep information seeking QA pairs based on the real-world web environment (§2.1); **Step** II: Sample high-quality trajectories from QA pairs using both LLMs and LRMs to guide the agency learning process (§2.2); **Step** III: Perform fine-tuning to adapt the format instruction following to agentic tasks and environments (§3.1); **Step** IV: Apply RL to optimize the agent's decision-making and generalization capabilities in real-world web environments (§3.2). We offer a systematic, end-to-end pipeline for building long-term information-seeking web agents.

Extensive experiments on two web information seeking benchmarks, GAIA and WebWalkerQA, show the effectiveness of our pipeline and WebDancer (§4). We further present a comprehensive analysis covering data efficiency, agentic system evaluation, and agent learning (§5).
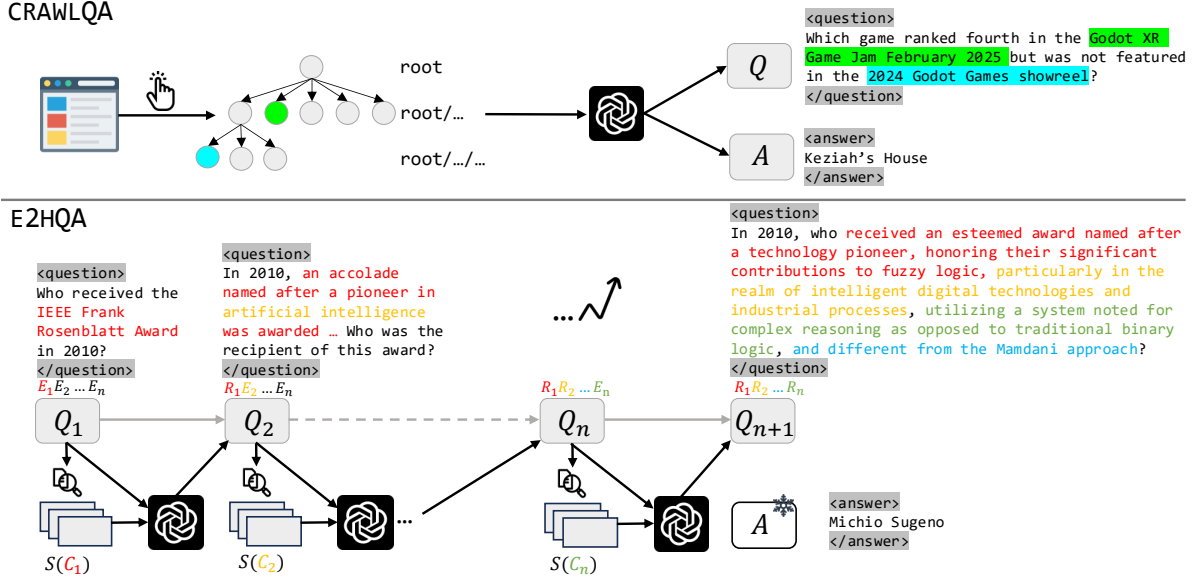
Figure 1: **Two web data generation pipelines.** ❶ For CRAWLQA, we first collect `root` url of knowlageable websites. Then we mimic human behavior by systematically clicking and collecting subpages accessible through sublinks on the `root/...` page. Using predefined rules, we leverage GPT4o to generate synthetic QA pairs based on the gathered information. ❷ For E2HQA, the initial question Q_1 is iteratively evolved using the new information C_i retrieved from the entity E_i at iteration i, allowing the task to progressively scale in complexity, from simpler instances to more challenging ones. We use GPT-4o to rewrite the question until the iteration reaches n.

2 Deep Information Seeking Dataset Synthesis

2.1 QA Pairs Construction

To enable longer-horizon web exploration trajectories, it is essential to curate a substantial corpus of complex and diverse QA pairs that can elicit multi-step reasoning, goal decomposition, and rich interaction sequences. The main requirements for these QAs are: *(i)* diversity of question types, and *(ii)* increased task complexity as measured by the number of interaction steps required for resolution. In contrast to prior datasets that predominantly involve shallow queries solvable in 2–3 steps, our objective is to scale both the volume and the depth of multi-hop reasoning. To achieve this, we primarily develop the below datasets: CRAWLQA and E2HQA.

CRAWLQA   Constructing QA pairs based on information crawled from web pages represents an effective paradigm for scalable knowledge acquisition Wu et al. (2025). We begin by collecting the root URLs of official and knowledgeable websites spanning arxiv, github, wiki, *etc.* Mialon et al. (2023) To emulate human browsing behavior, we recursively navigate subpages by following accessible hyperlinks from each root site. We employ GPT-4o to synthesize QA pairs from the collected content. To ensure specificity and relevance of questions, inspired by Sen et al. (2022), we prompt LLMs to generate questions of designed types (*e.g.,* COUNT, MULTI-HOP, INTERSECTION) via in-context learning Brown et al. (2020).

E2HQA   Similar to the reverse construction strategy Wei et al.; Zhou et al. (2025), we begin from large QA pairs in SimpleQA style OpenAI (2025b) where each answer is a concise, fact-seeking entity. We first select an entity E_n from the question Q_n, where n represents the number of refinement iterations. Then, we use the LLMs to construct a query based on this entity in order to search via search engine S for information C_n related to E_n. After that, we use LLMs π to restructure the obtained content into a new query R_n to replace the original entity in the question. The process can be signaled as: $R_n = \pi(S(C_n))$. This way, the new question Q_{n+1} requires solving the sub-problem we have constructed before finding

the answer to the original question. Moreover, it ensures that the answer does not change during the question refinement, thereby preserving the validity of the QA pairs. By continuously searching, we can gradually rephrase an initially simple question into a more complex multi-step one. Moreover, the number of steps needed to solve the problem can be controlled by adjusting the number of rephrasing times.

2.2 Agent Trajectories Rejection Sampling

Agent Setup Our agent framework is based on `ReAct` Yao et al. (2023), the most popular approach to language agents. A `ReAct` trajectory consists of multiple `Thought`-`Action`-`Observation` rounds, where an LM generates free-form `Thought` for versatile purposes, and structured `Action` to interact with environments (tools) and receive `Observation` feedback. We assume that the agent execution loop at time t can be denoted as (τ_t, α_t, o_t), where τ denotes `Thought`, α signifies `Action`, and o represents `Observation`. α can be further expressed as (α^m, α^p), where α^m is the name of the action, and α^p is the parameters required to perform the action. $\alpha^m \in \{search, visit, answer\}$, which corresponds to the two most important agentic tools in the deep information seeking. For *search* 🔍 `action`, α^p consists of *query* and *filter_year*, while for *visit* 🖥 `action`, α^p consists of *goal* and *url_link*. The `observation` of *search* `action` includes the Top-10 titles and snippets, whereas the `observation` of the *visit* `action` is the *evidence* and *summary*, generated by a summarizer model M_s. The iteration terminates when the `action` is *answer*.

Then the historical trajectory can be signaled as:

$$\mathcal{H}_t = (\tau_0, \alpha_0, o_0, \tau_1, ..., \tau_{t-1}, \alpha_{t-1}, o_{t-1}). \tag{1}$$

At time step t, the agent receives an `observation` o_t from the web environment and generates `thought` τ_t taking an `action` α_t, following poliy $\pi(\tau_t, \alpha_t | \mathcal{H}_t)$.

The Chain-of-Thought (CoT) method has significantly enhanced the inferential capabilities of LLMs through a step-by-step reasoning process Wei et al. (2022), corresponding to the `thought` component in agentic systems. This process is critical for agentic execution, enabling high-level workflow planning, self-reflection, information extraction, adaptive action planning, and accurate `action` (tool usage).

Short and Long CoT Construction Agent models internalise the CoT generation capability as an active behavioral component of the model. Zhang et al. (2025e); Mai et al. (2025) The length of CoT and the associated thinking patterns play a crucial role in performance Team (2025a); Guo et al. (2025); Wu et al. (2024) We propose two simple yet effective methods for constructing the short CoT and long CoT, respectively. For short CoTs, we directly leverage the `ReAct` framework to collect the trajectories using a powerful model, GPT-4o. For long CoTs, we sequentially provide the LRMs, QwQ-Plus, with the historical `actions` and `observations` at each step, enabling it to decide the next `action` autonomously. Notably, we **exclude** the previous `thought` during further inference, as the LRM, QwQ-Plus, has not been exposed to multi-step reasoning inputs during training. However, we retain the `thought` at each step in the generated trajectory, as they serve as valuable supervision signals. The LRM's intermediate reasoning process, denoted as, denoted as "*<reasoning_content>*", is recorded as the current `thought` of the current step. Each constructed QA instance undergoes rejection sampling up to N times to ensure quality and coherence.

Trajectories Filtering We adopt a three-stage funnel-based trajectory filtering framework consisting of *validity control*, *correctness verification*, and *quality assessment*.

- For *validity control*, directly prompting LLMs to generate responses in the `ReAct` format under long-content conditions may result in non-compliance with instructions. In such cases, we discard these data points.
- For *correctness verification*, we only retain correct results. We follow the evaluation methodology proposed by Phan et al. (2025) and Wei et al. and use GPT-4o for accurate judgment.

- For *quality assessment*, we first apply rules to filter out trajectories with more than two actions, ensuring that there are no hallucinations and no severe repetitions. Subsequently, we filter the trajectories based on prompting to retain those that meet the following three criteria: Information Non-redundancy, Goal Alignment, and Logical Reasoning and Accuracy.

The QA pairs that are not present in the SFT dataset can be utilized during the reinforcement learning stage effectively. [1]

3 Multi-Step Multi-Tool Agent Learning

After obtaining high-quality trajectories in `ReAct` format, we seamlessly incorporate them into our agent SFT training stage. Specifically, `Thought` segments are closed by `<think>` and `</think>`, `Action` segments by `<tool_call>` and `</tool_call>`, `Observation` segments by `<tool_response>` and `</tool_response>`. The final `Action` segment corresponds to the final answer, enclosed by `<answer>` and `</answer>`. In addition, the QA data without trajectories, which those filtered during earlier stages, can be effectively leveraged during the RL phase. We first train a policy model π_θ via agent SFT for cold start, followed by agent RL for generalization. The overall training framework is illustrated in Figure 2.
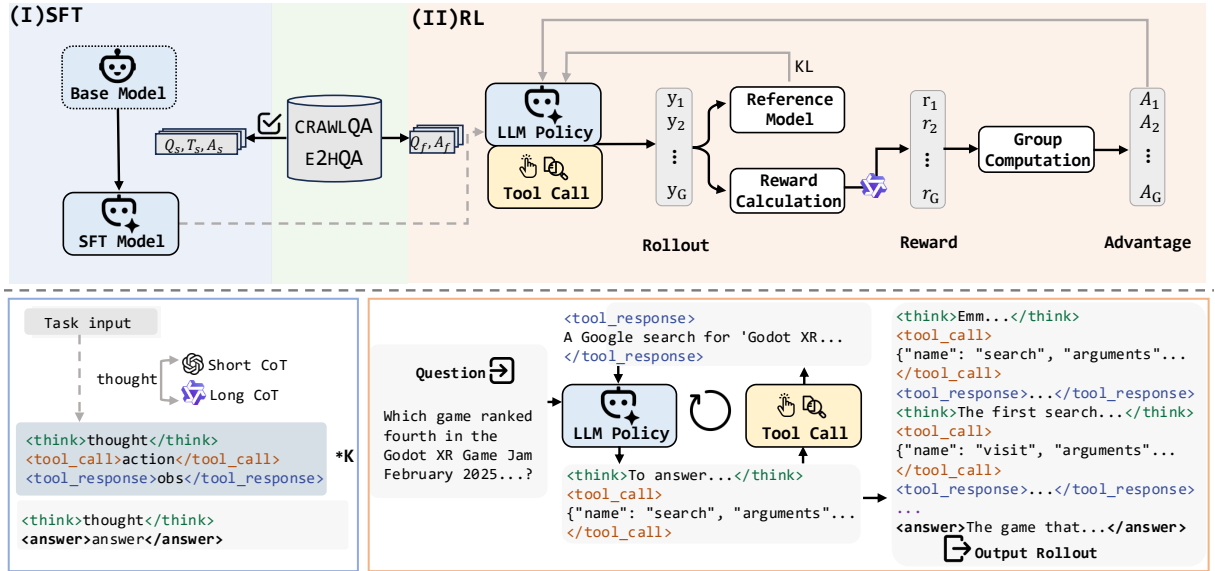

Figure 2: **The overview of training framework**. (I) The SFT stage for cold start utilizes the reformatted `ReAct` datasets, where the thought includes both short and long CoT, respectively. (II) The RL stage performs rollouts with the tool calls on the QA pairs that are not utilized during the SFT stage, and optimizes the policy using the DAPO algorithm.

3.1 Agent Supervised Fine Tuning

To capture complete agentic trajectories, we train the policy model θ via supervised fine-tuning on obtained decision-making trajectories. The cold start enhances the model's capability to couple multiple reasoning and action steps, teaching it a behavioral paradigm of alternating reasoning with action, while preserving its original reasoning capabilities as much as possible. Following the empirical findings of Chen et al. (2023; 2025); Zhang et al. (2025e), to avoid interference from external feedback during learning, we mask out loss contributions from `observation` in the agentic world modelling task, which has been shown to generally improve performance and robustness. Given the task context **tc** and the complete agentic execution trajectory $\mathcal{H} = (x_0, x_1, ..., x_{n-1}, x_n)$, where each $x_i \in \{\tau, \alpha, o\}$, the loss function L is

---

[1]The details of training datasets and are shown in App. D.

computed as follows:

$$L = -\frac{1}{\sum_{i=1}^{|\mathcal{H}|} \mathbb{I}[x_i \neq o]} \sum_{i=1}^{|\mathcal{H}|} \mathbb{I}[x_i \neq o] \cdot \log \pi_\theta(x_i \mid \mathbf{tc}, x_{<i}) \tag{2}$$

Here, $\mathbb{I}[x_i \neq o]$ filters out tokens corresponding to external feedback, ensuring that the loss is computed over the agent's autonomous decision steps. The SFT stage offers strong initialization for the subsequent RL stage Zhang et al. (2025b).

3.2 Agent Reinforcement Learning

The agent RL stage aims to internalize the agency capability into the reasoning model, enhancing its multi-turn, multi-tool usage capacity with outcome-based rewards. Kumar et al. (2025) Building on the SFT stage, RL employs Decoupled Clip and Dynamic Sampling Policy Optimization algorithm to refine and incentivize the policy model π_θ's ability to interleave `Thought`-`Action`-`Observation` sequences. **DAPO** Decoupled Clip and Dynamic Sampling Policy Optimization (**DAPO**) algorithm is an RL algorithm that optimizes a policy π_θ to produce higher-reward outputs under a reward model R Yu et al. (2025); Ferrag et al. (2025). For each question-answer pair (q, a) from the data distribution $\mathcal{D}$, DAPO samples a set of candidate agentic executions $\{o_i\}_{i=1}^G$. The policy is then updated to maximize the following objective:

$$\mathcal{J}_{\text{DAPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^G\sim\pi_{\theta_{\text{old}}}(\cdot|context)}$$

$$\left[\frac{1}{\sum_{i=1}^G |o_i|} \sum_{i=1}^G \sum_{t=1}^{|o_i|} \min\left(r_{i,t}(\theta)\hat{A}_{i,t}, \, \text{clip}\left(r_{i,t}(\theta), 1 - \varepsilon_{low}, 1 + \varepsilon_{high}\right)\hat{A}_{i,t} \right) \right] \tag{3}$$

$$\text{s.t.} \quad 0 < \left| \{o_i \mid \mathbf{is\_equivalent}(y, o_i)\} \right| < G,$$

where agentic execution o_i refers solely to the tokens generated by models, excluding any tool responses. In contrast, *context*, including both the model outputs and tool responses, is used to construct the input trajectory for computing $\pi_{\theta_{\text{old}}}$. However, the optimization is applied only to the model-generated portion o_i, aligning with the SFT. ε is the clipping range of the importance sampling ratio $r_{i,t}(\theta)$. And $\hat{A}_{i,t}$ is an estimator of the advantage of the i-th agentic executions at time step t:

$$r_{i,j}(\theta) = \frac{\pi_\theta(o_i \mid q_i, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_i \mid q_i, o_{i,<t})}, \quad \hat{A}_{i,j} = \frac{R_i - \text{mean}(\{R_i\})}{\text{std}(\{R_i\})}, \tag{4}$$

The dynamic sampling mechanism over-samples and filters out prompts with accuracy equal to 1 and 0. It is crucial in our data-training pipeline, as the remaining QA pairs, being synthetically generated—may contain invalid or noisy instances that could otherwise degrade policy learning. Such unreliable samples can be effectively ignored, ensuring the agent focuses on learning from high-quality signals.

Agentic Action Rollout Within the `ReAct` framework, each round of agentic execution begins by generating a `thought`, closed by `<think>` and `</think>`, followed by a `action` name α^m and corresponding parameters α^p, enclosed by `<tool_call>` and `</tool_call>` operation, all conditioned on the iteration history $\mathcal{H}$. These components are iteratively used to interact with the real-world search environment, producing an `observation` as feedback, bounded by `<tool_response>` and `</tool_response>` upon the `<tool_response>` is detected. The round of interaction spans from `<think>` to `</tool_response>`. The rollout concludes with the generation of `<answer>` and `</answer>`, following the final `thought`.

Reward Design The reward design plays a critical role during the RL training process Guo et al. (2025). Our reward system mainly consists of two types of rewards, $score_{\text{format}}$ and $score_{\text{answer}}$. Given that format consistency has been largely addressed during the initial RFT stage, we assign a small weight to the $score_{\text{format}}$ in the overall reward. The $score_{\text{format}}$ is binary: it is set to 1 only if the entire output strictly conforms to the required format and all tool calls in *json* format are valid. Considering that the QA

answers are inherently non-verifiable, cannot be reliably evaluated using rule-based F1/EM metrics, despite the brevity of the responses, and that the final evaluation relies on *LLM-as-Judge* Zheng et al. (2023) which the judge model is M_j, we opt to employ model-based prompt evaluation as the answer reward signal Seed et al. (2025); Xu et al. (2025); Liu et al. (2025b). The $score_{answer}$ is also binary, assigned as 1 only when the response is judged as correct by the LLMs. The final reward function is:

$$R(\hat{y}_i, y) = 0.1 * score_{format} + 0.9 * score_{answer} \tag{5}$$

where $\hat{y}_i$ denotes the model prediction and y is the reference answer.

4 Experiments

Table 1: **Main results** on GAIA and WebWalkerQA benchmarks. We discuss the reported results of baselines and concurrent works in App. C.1. "-" means results that are either not reproducible or not reported. The best results among all frameworks are in **bolded**.

Backbone	Framework	GAIA				WebWalkerQA			
		Level 1	Level 2	Level 3	Avg.	Easy	Medium	Hard	Avg.
No Agency									
Qwen-2.5-7B	Base	12.8	3.8	0.0	6.8	1.25	0.8	0.7	0.8
Qwen-2.5-32B	Base	20.5	9.6	8.3	13.6	3.8	2.5	3.3	3.1
	RAG	12.8	11.8	8.3	11.8	23.1	14.3	11.3	15.3
Qwen-2.5-72B	Base	20.5	13.5	0.0	14.6	9.4	7.1	3.3	6.3
GPT-4o	Base	23.1	15.4	8.3	17.5	6.7	6.0	4.2	5.5
QwQ-32B	Base	30.8	15.4	25.0	22.3	7.5	2.1	4.6	4.3
	RAG	33.3	36.5	8.3	32.0	36.9	26.1	33.5	31.2
DeepSeek-R1-671B	Base	43.6	26.9	8.3	31.1	5.0	11.8	11.3	10.0
Close-Sourced Agentic Frameworks									
	OpenAI DR	74.3	69.1	47.6	67.4	-	-	-	-
Open-sourced Agentic Frameworks									
Qwen-2.5-7B	Search-o1	23.1	17.3	0.0	17.5	-	-	-	-
	R1-Searcher	28.2	19.2	8.3	20.4	-	-	-	-
Qwen-2.5-32B	Search-o1	33.3	25.0	0.0	28.2	-	-	-	-
QwQ-32B	Search-o1	53.8	34.6	16.7	39.8	43.1	35.0	27.1	34.1
	WebThinker-Base	53.8	44.2	16.7	44.7	47.2	41.1	39.2	41.9
	WebThinker-RL	56.4	50.0	16.7	48.5	58.8	44.6	40.4	46.5
	Simple DS	-	-	-	50.5	-	-	-	-
ReAct Agentic Frameworks									
Qwen-2.5-7B	Vanilla `ReAct`	28.2	15.3	0.0	18.4	28.1	31.2	16.0	24.2
	WebDancer	41.0	30.7	0.0	31.0	40.6	44.1	28.2	36.0
Qwen-2.5-32B	Vanilla `ReAct`	46.1	26.9	0.0	31.0	35.6	38.7	22.5	31.9
	WebDancer	46.1	44.2	8.3	40.7	44.3	46.7	29.2	38.4
QwQ-32B	Vanilla `ReAct`	48.7	34.6	16.6	37.8	35.6	29.1	13.2	24.1
	WebDancer	**61.5**	**50.0**	**25.0**	**51.5**	**52.5**	**59.6**	**35.4**	**47.9**
GPT-4o	Vanilla ReAct	51.2	34.6	8.3	34.6	34.6	42.0	23.9	33.8

4.1 Experimental Setup

We evaluate our approach on two established deep information-seeking benchmarks: **GAIA** and **Web-WalkerQA**. In this work, we adopt the *LLM-as-Judges* paradigm to evaluate both tasks using the `Pass@1` metric, following Team (2025b). The details of the datasets and baselines are introduced in App. E.1 and App. E.2, respectively. The implementation details are shown in App. E.3. Qwen-7B and Qwen-32B are

trained on Short-CoT datasets, while QwQ-32B is trained on Long-CoT datasets. Further analyses are shown in Sec. 5.

4.2 Experimental Results

Main Results As shown in Table 1, frameworks without agentic capabilities (*No Agency*) perform poorly on both the GAIA and WebWalkerQA benchmarks, highlighting the necessity of active information-seeking and agentic decision-making for these tasks. The closed-source agentic system, *OpenAI DR*, through end-to-end RL training achieves the highest scores. Among Open-sourced frameworks, agentic approaches built on top of native strong reasoning models like QwQ-32B consistently outperform their non-agentic counterparts, demonstrating the effectiveness of leveraging reasoning-specialized models in agent construction. Importantly, under the highly extensible `ReAct` framework, our proposed **WebDancer** shows substantial gains over the vanilla `ReAct` baseline across different model scales. Notably, it even surpasses the performance of GPT-4o in the best-case scenario. This demonstrates that even within a lightweight framework, our method significantly enhances agentic capabilities over the underlying base model, validating the strength and generality of our approach.

Results on More Challenging Benchmarks We evaluate our approach on two more challenging datasets, BrowseComp (*En.*) Wei et al. and BrowseComp-zh (*Zh.*) Zhou et al. (2025), which are designed to better reflect complex information-seeking scenarios using `PASS@1/PASS@3`. As shown in Table 2, **WebDancer** demonstrates consistently strong performance across both datasets, highlighting its robustness and effectiveness in handling difficult reasoning and information-seeking tasks.

Table 2: Results on BrowseComp (*En.*) and BrowseComp-zh (*Zh.*).

Framework	Browsing		*En.*	*Zh.*
GPT-4o	✘		0.6	6.2
	✔		1.9	-
QwQ-32B	✘		-	11.1
WebDancer	✔		3.8/**7.9**	18.0/**31.5**

5 Analysis

Detailed Results We conduct detailed analyses on the GAIA datasets. Given the dynamic and complex nature of agent environments, as well as the relatively small and variable test set, we further conduct a fine-grained analysis of `Pass@3` and `Cons@3` in Figure 4. The `Cons@3` metric is computed by evaluating the number of correct responses out of three independent attempts: achieving one correct answer yields a score of 1/3, two correct answers yield 2/3, and three correct answers result in a full score of 1. For non-reasoning models, RL leads to substantial improvements in both `Pass@3` and `Cons@3`. Notably, the `Pass@1` performance after RL is comparable to the `Pass@3` of the SFT baseline, consistent with previous findings Yue et al. (2025); Swamy et al. (2025) suggesting that RL can sample correct responses more efficiently. For LRMs, while the improvements in `Pass@1`, `Pass@3`, and `Cons@3` after RL are marginal, a noticeable gain in consistency is observed; this may be due to sparse reward signals caused by excessively long trajectories Feng et al. (2025); Wei et al. (2025b). This suggests that continued on-policy optimization may yield limited benefits for LRMs in agentic tasks. **Our best-performing model achieves a `Pass@3` score of 64.1% on GAIA and 62.0% on WebWalkerQA**.

High-quality trajectory data is crucial for effective SFT of agents. We propose two data construction strategies, resulting in the creation of datasets CRAWLQA and E2HQA. After applying trajectory rejection sampling to the QA data, we further perform filtering to enhance data quality. In Figure 3, we conduct ablation studies on the QwQ and evaluate the
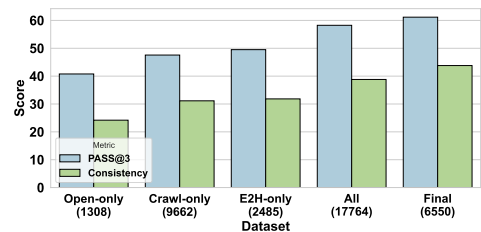

Figure 3: Results on data efficiency using GAIA benchmark. Open-only refers to using only challenging QA datasets from open-source sources.
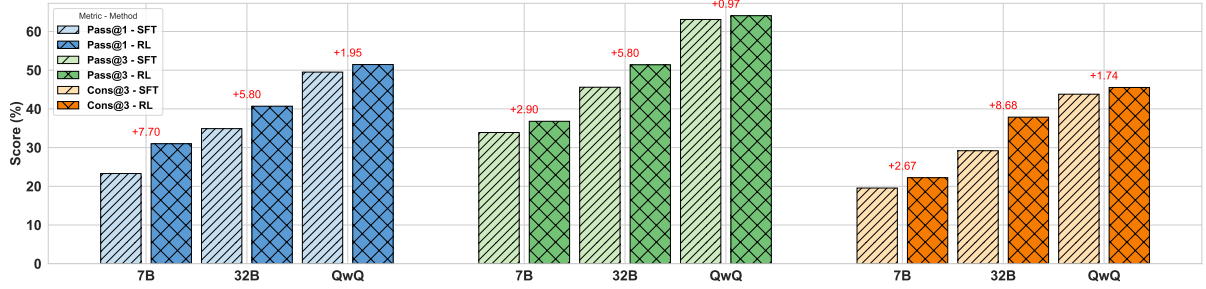
Figure 4: Detailed evaluation results using `Pass@1`, `Pass@3` and `Cons@3` metric on GAIA benchmark.

effectiveness of the constructed datasets. In long-CoT, hallucinations often arise when the model attempts to answer by simulating observations, primarily due to its exclusive reliance on internal reasoning mechanisms. Li et al. (2025a) Final performs better than all under low-data regimes, emphasizing the value of robust filtering.

SFT for cold start is essential, as the agent tasks demand strong multi-step multi-tool instruction-following capabilities. We empirically investigate this by comparing performance under a single reinforcement learning setting using QwQ. The results show that the `Pass@3` performance is significantly limited, achieving **only 5%** on the GAIA. For the RL phase, both `Pass@3` and `Cons@3` show consistent improvements as the number of training steps increases, as illustrated in Figure 5(a).

Table 3: Results on CoT knowledge transfer. *Inv.* denotes invalid rate. **R.** refers to whether the model is a reasoning model.

| Model	R.	Short-Cot			Long-Cot		
		Pass@3	Cons@3	*Inv.*	Pass@3	Cons@3	*Inv.*
Qwen2.5-7B	✗	33.98	22.33	0.65%	35.92	21.00	21.36%
Qwen2.5-32B	✗	42.72	24.33	4.20%	45.63	30.00	13.59%
QwQ-32B	✔	44.66	28.33	0.97%	58.25	39.66	13.27%

The thinking pattern knowledge used by strong reasoner models is struggle transferable to those of small instruction models. As shown in Table 3, reasoning models trained on trajectories synthesized by reasoning models significantly enhance their reasoning performance Gou et al. (2023). For non-reasoning models, Long-CoT also demonstrates good performance, but it introduces additional issues, such as a higher invalid rate, often manifested as `repetition`, leading to exceeding the model's context length, particularly in smaller-scale models. These reasoning patterns do not easily transfer to instruction-tuned models, which are generally optimized for task-following behavior rather than deep reasoning. This observation aligns with the findings in Li et al. (2025b); Yin et al. (2025), which highlight the brittleness of cross-model reasoning knowledge transfer. [2] As such, direct transfer of reasoning capabilities from reasoner models to instruction models remains a non-trivial challenge.

RL enables longer reasoning processes and supports more complex agentic action. As demonstrated by the results on Qwen-32B in Figure 5(b), we observe that SFT leads to more frequent action generation and extended reasoning sequences, largely due to the nature of our training data (App. E.1). RL frameworks facilitate the emergence of more sophisticated reasoning strategies by allowing models to optimize over sequences of decisions, rather than single-step outputs. This enables models to learn from delayed rewards and engage in deeper exploration of action spaces, leading to more coherent and longer reasoning trajectories. Moreover, RL encourages agentic behaviors where models autonomously decide intermediate steps, subgoals, or tools to achieve final objectives, as shown in App. F. Such capabilities are particularly useful in complex environments where straightforward task-following fails to generalize.

Web agent executes in a dynamic, evolving environment that inherently resists stabilization. As shown in Figure 5(c), adjusting the decoding temperature had minimal impact on final performance, indicating that decoding variability alone does not account for agent instability. Instead, we attribute much of

---

[2]We also experiment with mixing short-CoTs and long-CoTs, but observe no significant performance improvements.
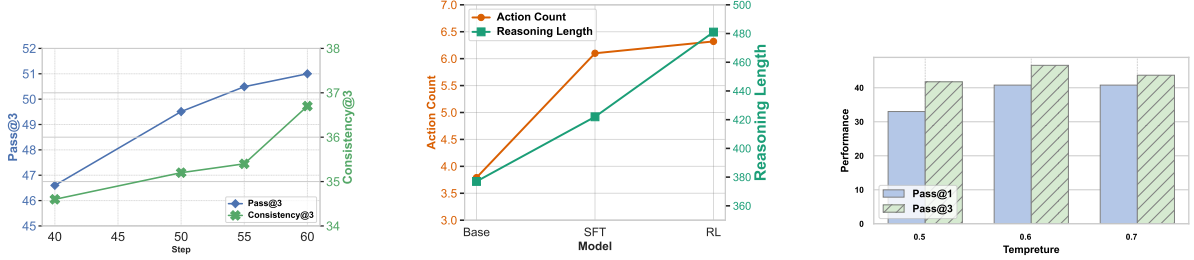
(a) Performance across training steps using the DAPO algorithm.

(b) Evolution of thought length and number of actions.

(c) Pass@1 and Pass@3 results on different temperatures.

Figure 5: Analysis on RL algorithm, emergent agency, and agent environments using GAIA benchmark.

the performance fluctuation to changes in the web environment itself, highlighting the non-stationary and open-ended nature of real-world agent deployment. Unlike static datasets with fixed distributional properties, real-world environments evolve over time, requiring agents to remain robust under changing contexts and partial observability. Additionally, to further investigate potential overfitting, we conduct a memorization stress test: we fine-tuned a Qwen-7B model on 69 correctly sampled trajectories from the GAIA development set for 10 epochs, and subsequently evaluate its performance on the same set. Despite this, greedy decoding **only achieved 37.4%**, suggesting the difficulty of stabilization on the open-domained agentic tasks.

6   Related Works

Information Seeking Agents and Benchmarks. Recent advances in information-seeking agents aim to integrate web interaction into LLMs' reasoning. Xi et al. (2025) WebThinker Team (2025b) and Search-o1 Li et al. (2025a) use tool-augmented LLMs that actively retrieve evidence mid-inference. Some works like R1-Searcher Song et al. (2025), ReSearch Chen et al. (2025) and Search-R1 Jin et al. (2025) focus on reinforcement learning to teach search behavior from outcome-based rewards. DeepResearcher Zheng et al. (2025) extends this by operating in real web environments with online RL, while SimpleDeepSearcher Sun et al. (2025) shows that a small number of distilled demonstrations can train effective agents without full RL. These works demonstrate promising capabilities but often rely on limited or simplistic data. In parallel, benchmarks like GAIA Mialon et al. (2023) and WebWalkerQA Wu et al. (2025) test reasoning and browsing, but many are single-turn or domain-limited. BrowseComp Wei et al. and BrowseComp-zh Zhou et al. (2025) increase task complexity, requiring multi-hop search and multilingual reasoning, yet still lack diversity and scalability. Our work addresses these gaps by proposing automatic synthesis QA datasets designed to challenge agents across domains and task types in more realistic web environments.

Agents Learning. Agent learning has evolved from in-context learning towards training-based methods Liu et al. (2025a); Zhou et al. (2024; 2023). Recent studies Qiao et al. (2024); Zeng et al. (2024); Chen et al. (2024) have primarily focused on leveraging SFT with curated task-solving trajectories following the `ReAct` paradigm. However, empirical evidence suggests that pure SFT-based agents often exhibit limited generalization performance when confronted with adaptive operational contexts Zheng et al. (2025); Zhang et al. (2025d); Qian et al. (2025); Yu et al. (2024). Building upon these limitations, RL-based methods Song et al. (2025); Zheng et al. (2025;?); Zhang et al. (2025d;c) have demonstrated remarkable potential in developing sophisticated search strategies through learned exploration policies. Despite their theoretical advantages, practical implementations face persistent challenges in training stability and sample efficiency. **WebDancer** implements a two-stage framework: an initial cold-start phase employing trajectory-based SFT to establish fundamental agency patterns, followed by targeted RL to cultivate adaptive long-term agency capabilities.

7 Conclusion

In this work, we propose a systematic framework for building end-to-end multi-step information-seeking web agents from scratch. By introducing scalable QA data synthesis methods and a two-stage training pipeline combining SFT and on-policy RL, our WebDancer agent achieves strong performance on GAIA and WebWalkerQA. These findings underscore the significance of our proposed training strategy and provide valuable insights into the critical aspects of agent training. Moving forward, this research offers actionable and systematic pathways for the community to advance the development of increasingly sophisticated agentic models capable of tackling complex real-world information-seeking tasks.

References

anthropic. Meet claude, 2025. URL https://www.anthropic.com/claude.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Shiyi Cao, Sumanth Hegde, Dacheng Li, Tyler Griggs, Shu Liu, Eric Tang, Jiayi Pan, Xingyao Wang, Akshay Malik, Graham Neubig, Kourosh Hakhamaneshi, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Skyrl-v0: Train real-world long-horizon agents via reinforcement learning, 2025.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023.

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

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. Agent-flan: Designing data and methods of effective agent tuning for large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pp. 9354–9366, 2024.

Runnan Fang, Xiaobin Wang, Yuan Liang, Shuofei Qiao, Jialong Wu, Zekun Xi, Ningyu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, et al. Synworld: Virtual scenario synthesis for agentic action knowledge refinement. *arXiv preprint arXiv:2504.03561*, 2025.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training, 2025. URL https://arxiv.org/abs/2505.10978.

Mohamed Amine Ferrag, Norbert Tihanyi, and Merouane Debbah. Reasoning beyond limits: Advances and open problems for llms. *arXiv preprint arXiv:2503.22732*, 2025.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. Tora: A tool-integrated reasoning agent for mathematical problem solving. *ArXiv*, abs/2309.17452, 2023. URL https://api.semanticscholar.org/CorpusID:263310365.

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

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.

Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Bowei Xia, Tao Sun, Ziyu Ye, Zhaoxuan Jin, Yingru Li, Zeyu Zhang, Yifeng Wang, Qianshuo Ye, Ping Luo, and Guohao Li. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation, 2025. URL https://github.com/camel-ai/owl.

Bowen Jin, Hansi Zeng, Zhenrui Yue, 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.

Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip HS Torr, Fahad Shahbaz Khan, and Salman Khan. Llm post-training: A deep dive into reasoning large language models. *arXiv preprint arXiv:2502.21321*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

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*, 2025a.

Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. Small models struggle to learn from strong reasoners. *arXiv preprint arXiv:2502.12143*, 2025b.

Xinbin Liang, Jinyu Xiang, Zhaoyang Yu, Jiayi Zhang, Sirui Hong, Sheng Fan, and Xiao Tang. Openmanus: An open-source framework for building general ai agents, 2025. URL https://doi.org/10.5281/zenodo.15186407.

Bang Liu, Xinfeng Li, Jiayi Zhang, Jinlin Wang, Tanjin He, Sirui Hong, Hongzhang Liu, Shaokun Zhang, Kaitao Song, Kunlun Zhu, et al. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems. *arXiv preprint arXiv:2504.01990*, 2025a.

Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. Inference-time scaling for generalist reward modeling. *arXiv preprint arXiv:2504.02495*, 2025b.

Xinji Mai, Haotian Xu, Xing W, Weinong Wang, Yingying Zhang, and Wenqiang Zhang. Agent rl scaling law: Agent rl with spontaneous code execution for mathematical problem solving, 2025.

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

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.

OpenAI. Gpt-4 system card, 2022. URL https://cdn.openai.com/papers/gpt-4-system-card.pdf.

OpenAI. Deep research system card, 2025a. URL https://cdn.openai.com/deep-research-system-card.pdf.

OpenAI. Introducing simpleqa, 2025b. URL https://openai.com/index/introducing-simpleqa/.

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

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

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*, 2025.

Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Jiang, Chengfei Lv, and Huajun Chen. Autoact: Automatic agent learning from scratch for qa via self-planning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3003–3021, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

ByteDance Seed, Yufeng Yuan, Yu Yue, Mingxuan Wang, Xiaochen Zuo, Jiaze Chen, Lin Yan, Wenyuan Xu, Chi Zhang, Xin Liu, et al. Seed-thinking-v1. 5: Advancing superb reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.13914*, 2025.

Priyanka Sen, Alham Fikri Aji, and Amir Saffari. Mintaka: A complex, natural, and multilingual dataset for end-to-end question answering. In Nicoletta Calzolari, Chu-Ren Huang, Hansaem Kim, James Pustejovsky, Leo Wanner, Key-Sun Choi, Pum-Mo Ryu, Hsin-Hsi Chen, Lucia Donatelli, Heng Ji, Sadao Kurohashi, Patrizia Paggio, Nianwen Xue, Seokhwan Kim, Younggyun Hahm, Zhong He, Tony Kyungil Lee, Enrico Santus, Francis Bond, and Seung-Hoon Na (eds.), *Proceedings of the 29th International Conference on Computational Linguistics*, pp. 1604–1619, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics. URL https://aclanthology.org/2022.coling-1.138/.

Yijia Shao, Yucheng Jiang, Theodore A Kanell, Peter Xu, Omar Khattab, and Monica S Lam. Assisting in writing wikipedia-like articles from scratch with large language models. In *NAACL-HLT*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

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.

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

Gokul Swamy, Sanjiban Choudhury, Wen Sun, Zhiwei Steven Wu, and J. Andrew Bagnell. All roads lead to likelihood: The value of reinforcement learning in fine-tuning, 2025. URL https://arxiv.org/abs/2503.01067.

QwQ Team. Qwq-32b: Embracing the power of reinforcement learning, 2025a. URL https://qwenlm.github.io/blog/qwq-32b/.

WebThinker Team. Webthinker: Empowering large reasoning models with deep research capability, 2025b. URL https://github.com/RUC-NLPIR/WebThinker. Github.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition, 2022. URL https://arxiv.org/abs/2108.00573.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I. Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution, 2025a.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, Hyokun Yun, and Lihong Li. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning, 2025b. URL https://arxiv.org/abs/2505.16421.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Deyu Zhou, Pengjun Xie, and Fei Huang. Webwalker: Benchmarking llms in web traversal, 2025. URL https://arxiv.org/abs/2501.07572.

Siwei Wu, Zhongyuan Peng, Xinrun Du, Tuney Zheng, Minghao Liu, Jialong Wu, Jiachen Ma, Yizhi Li, Jian Yang, Wangchunshu Zhou, Qunshu Lin, Junbo Zhao, Zhaoxiang Zhang, Wenhao Huang, Ge Zhang, Chenghua Lin, and J. H. Liu. A comparative study on reasoning patterns of openai's o1 model, 2024. URL https://arxiv.org/abs/2410.13639.

x.ai. Grok 3 beta — the age of reasoning agents, 2025. URL https://x.ai/news/grok-3.

Zekun Xi, Wenbiao Yin, Jizhan Fang, Jialong Wu, Runnan Fang, Ningyu Zhang, Jiang Yong, Pengjun Xie, Fei Huang, and Huajun Chen. Omnithink: Expanding knowledge boundaries in machine writing through thinking. *arXiv preprint arXiv:2501.09751*, 2025.

Wenyuan Xu, Xiaochen Zuo, Chao Xin, Yu Yue, Lin Yan, and Yonghui Wu. A unified pairwise framework for rlhf: Bridging generative reward modeling and policy optimization. *arXiv preprint arXiv:2504.04950*, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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, 2018. URL https://arxiv.org/abs/1809.09600.

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.

Huifeng Yin, Yu Zhao, Minghao Wu, Xuanfan Ni, Bo Zeng, Hao Wang, Tianqi Shi, Liangying Shao, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. Towards widening the distillation bottleneck for reasoning models, 2025.

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

Yuanqing Yu, Zhefan Wang, Weizhi Ma, Zhicheng Guo, Jingtao Zhan, Shuai Wang, Chuhan Wu, Zhiqiang Guo, and Min Zhang. Steptool: A step-grained reinforcement learning framework for tool learning in llms. *arXiv preprint arXiv:2410.07745*, 2024.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?, 2025. URL https://arxiv.org/abs/2504.13837.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. In *Findings of the Association for Computational Linguistics ACL 2024*, pp. 3053–3077, 2024.

Chaoyun Zhang, Shilin He, Liqun Li, Si Qin, Yu Kang, Qingwei Lin, and Dongmei Zhang. Api agents vs. gui agents: Divergence and convergence. *arXiv preprint arXiv:2503.11069*, 2025a.

Chong Zhang, Yue Deng, Xiang Lin, Bin Wang, Dianwen Ng, Hai Ye, Xingxuan Li, Yao Xiao, Zhanfeng Mo, Qi Zhang, et al. 100 days after deepseek-r1: A survey on replication studies and more directions for reasoning language models. *arXiv preprint arXiv:2505.00551*, 2025b.

Dingchu Zhang, Yida Zhao, Jialong Wu, Baixuan Li, Wenbiao Yin, Liwen Zhang, Yong Jiang, Yufeng Li, Kewei Tu, Pengjun Xie, and Fei Huang. Evolvesearch: An iterative self-evolving search agent, 2025c. URL https://arxiv.org/abs/2505.22501.

Shaokun Zhang, Yi Dong, Jieyu Zhang, Jan Kautz, Bryan Catanzaro, Andrew Tao, Qingyun Wu, Zhiding Yu, and Guilin Liu. Nemotron-research-tool-n1: Tool-using language models with reinforced reasoning. *arXiv preprint arXiv:2505.00024*, 2025d.

Yuxiang Zhang, Yuqi Yang, Jiangming Shu, Xinyan Wen, and Jitao Sang. Agent models: Internalizing chain-of-action generation into reasoning models. *arXiv preprint arXiv:2503.06580*, 2025e.

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

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, 2025. URL https://arxiv.org/abs/2504.03160.

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

Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang, Jing Chen, Ruipu Wu, Shuai Wang, Shiding Zhu, Jiyu Chen, Wentao Zhang, Xiangru Tang, Ningyu Zhang, Huajun Chen, Peng Cui, and Mrinmaya Sachan. Agents: An open-source framework for autonomous language agents. 2023. URL https://arxiv.org/abs/2309.07870.

Wangchunshu Zhou, Yixin Ou, Shengwei Ding, Long Li, Jialong Wu, Tiannan Wang, Jiamin Chen, Shuai Wang, Xiaohua Xu, Ningyu Zhang, Huajun Chen, and Yuchen Eleanor Jiang. Symbolic learning enables self-evolving agents. 2024. URL https://arxiv.org/abs/2406.18532.

Yuxin Zuo, Kaiyan Zhang, Shang Qu, Li Sheng, Xuekai Zhu, Biqing Qi, Youbang Sun, Ganqu Cui, Ning Ding, and Bowen Zhou. Ttrl: Test-time reinforcement learning, 2025. URL https://arxiv.org/abs/2504.16084.

A   Limitations

Although our proposed framework has demonstrated promising results, several limitations remain, which point to ongoing efforts and potential directions for future work.

Tool Number and Type   Currently, we integrate only two basic information-seeking tools. To enable more advanced and fine-grained retrieval capabilities, we plan to incorporate more sophisticated tools, such as *browser modeling* by abstracting browser functionalities into modular tools, and a *Python* sandbox environment for interacting with external APIs Wei et al. (2025a); Cao et al. (2025); Zhang et al. (2025a). This allows the agent to perform more human-like and efficient interactions, paving the way not only for tackling more challenging benchmarks but also for progressing toward more general and autonomous agency.

Task Generalization and Benchmarks   Our current experiments focus on two short-answer information-seeking tasks. However, a comprehensive web agent should also be capable of document-level research and generation Shao et al. (2024). Extending to such open-domain, long-form writing poses significant challenges in *reward modeling* in agentic tasks, which we are actively investigating, particularly how to design more reliable and informative reward signals for long-form generation in open-ended settings Liu et al. (2025b).

Data Utilization   While we have accumulated a large corpus of QA pairs and corresponding trajectories, effectively scaling learning remains a challenge, particularly in the RL stage, where only a small subset (*e.g.*, 5,000 pairs) can be utilized due to computational and stability constraints of RL in agentic tasks. This underscores the need for more efficient data utilization strategies to fully exploit the richness of the collected dataset.

High Rollout Cost   The RL phase incurs substantial computational and time overhead, as each rollout involves multiple rounds of tool invocations and LLM completions. This high cost not only limits scalability but also slows down iterative development and experimentation. A promising direction is to develop more efficient mechanisms for integrating tool calls with model completions, which can reduce rollout time and cost without sacrificing learning policy.

Hybrid Thinking   We consider two types of datasets characterized by short and long CoTs. Currently, our models are trained on a single dataset type. In future work, we plan to develop a hybrid reasoning agent model capable of dynamically controlling the reasoning length of the agent. Yang et al. (2025)

Thinking Pattern   In tool invocation, hallucinations may occur. For example, when dealing with mathematical problems, one might erroneously invoke a "*calculate*" tool that does not actually exist. Additionally, over-action may arise during the reasoning process, where redundant actions are performed even after the answer has been confirmed.

B   Broader Impacts

Building open-source, autonomous web agents capable of long-term information seeking has the potential to greatly benefit scientific research, education, and productivity by democratizing access to complex web-based reasoning tools. However, such systems also raise concerns, including the risk of misinformation propagation if agents rely on unreliable sources, and the possibility of misuse in automated content extraction or surveillance. We emphasize the importance of transparency, source attribution, and responsible deployment practices to mitigate potential harms.

C   Discussions

C.1 Concurrent Work

Comparison with the Training-based Methods   We primarily compare our approach with two training-based methods: WebThinker and SimpleDeepSearcher, highlighting the key differences. WebThinker also adopts an SFT followed by RL setup, but employs an off-policy RL algorithm Rafailov et al. (2023). Furthermore, WebThinker triggers actions and observations within the *<thinking_content>*, whereas our approach adopts a native `ReAct` style architecture, executing each action after completing its corresponding reasoning step. In contrast, Simple DeepSearcher relies solely on supervised fine-tuning over a carefully curated dataset. Our approach similarly follows an SFT-then-RL paradigm, but crucially leverages on-policy RL via DAPO. Our core contribution lies in **building a scalable end-to-end pipeline, from data construction to algorithmic design**, that supports native `ReAct` reasoning. This framework is compatible with both instruction LLMs and LRMs, enabling seamless integration and improved generalization.

Comparison with the Prompting-based Methods   Recent efforts in the community have explored building more autonomous and general-purpose agent systems, such as OWL Hu et al. (2025); Li et al. (2023), and OpenManus Liang et al. (2025), by leveraging foundation models with strong native agentic capabilities, such as Claude anthropic (2025). These systems typically rely on carefully engineered agent frameworks and prompting workflows, often involving multi-step tool usage and human-curated task structures. In contrast, we advocate for open-source models with emergent agency, crucial for democratizing agentic AI and advancing fundamental understanding of how agency can arise and scale in open systems. Our native `RAct` framework embraces simplicity, embodying the principle that less is more. **Training native agentic models is fundamentally valuable.**

C.2 Post-train Agentic Models

Agentic models refer to foundation models that natively support reasoning, decision-making, and multi-step tool use in interactive environments. They exhibit emergent capabilities such as planning, self-reflection, and action execution through structured prompting alone. Recent systems like *DeepSearch* and *Deep Research* illustrate how powerful foundation models can serve as agentic cores, enabling autonomous web interaction through native support for tool invocation and iterative reasoning. However, since web environments are inherently dynamic and partially observable, **reinforcement learning plays a crucial role in improving the agent's adaptability and robustness**. In this work, we aim to elicit autonomous agency in open-source models through targeted post-training.

D   Training Dataset

We collect 40K samples of **E2HQA** and 60K samples of CRAWLQA. These data samples are used to generate trajectories via either QwQ or GPT-4o, followed by a multi-stage filtering process to ensure quality, as described in Sec. 2.2. Table 4 separately reports the statistics for SFT data generated using Long-CoT and Short-CoT reasoning. We plan to scale this high-quality dataset further to investigate whether increasing the data volume leads to significant performance gains in future work.

Table 4: Statistics of training datasets. The thinking length is the average of the tokenized length of the thoughts.

CoT Type	Num.	Action Count	Thinking Length
Short	7,678	4.56	510.03
Long	6,550	2.31	1599.39

Filtering Criterion: Regarding the trajectory filter employed in Sec. 2.2, it is important to note that, during the quality assessment phase, we mitigate the presence of repetitive patterns by identifying and constraining the maximum occurrence of n-grams (n=10) within each trajectory to a threshold of 4. The purpose of this is to prevent the model from internalizing detrimental patterns, thereby safeguarding the integrity of the inference process.

Open-only Datasets: We select a set of widely-used QA datasets, including MuSiQue Trivedi et al. (2022), Bamboogle Press et al. (2022), PopQA Mallen et al. (2022), 2Wiki Ho et al. (2020), and HotpotQA Yang et al. (2018). To ensure question difficulty, we apply a simple RAG-based filtering process to remove easy questions.

E   Experimental Details

E.1   Benchmarks

GAIA is designed to evaluate general AI assistants on complex information retrieval tasks, while Web-WalkerQA focuses specifically on deep web information retrieval. Our experiments use 103 questions from GAIA's text-only validation split and 680 questions from the WebWalkerQA test set.

E.2   Baselines

We compare WebDancer against the following frameworks:

- *No Agency*: which denotes direct use base ability of models and simply uses retrieval-augmented generation (RAG). Includes Qwen2.5-7/32/72B-Instruct Yang et al. (2024), QwQ-32B Team (2025a), DeepSeek-R1-671B Guo et al. (2025), GPT-4o OpenAI (2022).

- *Close-Sourced Agentic Frameworks*: *OpenAI Deep Research (**DR**)* use end-to-end reinforcement learning to complete multitask research tasks.

- *Open-Sourced Agentic Frameworks*: **WebThinker** equips an LRM with a Deep Web Explorer to autonomously search and browse web pages mid-reasoning, interleaving tool use with chain-of-thought. For a fair comparison, we reproduced the results using Google Search and further replicated both the Base and RL versions of the method. **Search-o1** Li et al. (2025a) performs information-seeking by first generating search queries, retrieving web documents, and then using an LLM to answer based on the retrieved content, without optimizing the search process itself. **R1-Searcher** Song et al. (2025) trains an LLM to learn when and how to search using outcome-based reinforcement learning, without any supervised demonstrations.

E.3   Implements Details

We train using the multi-turn *chatml* format, structuring each dialogue such that tool responses are represented as user messages, and both thoughts and actions generated by the model are represented as assistant messages.

- **Dataset Construction:** The number of reject sampling $N = 5$. The summarizer model M_s is Qwen-2.5-72B. We build our system using the widely adopted `ReAct` framework, implemented on top of the Qwen-Agents [3].

- **Training and Inference:** We construct the judge model M_j based on Qwen-72B-Instruct, and design the reward prompt following Phan et al. (2025). For RL, we implement verl Sheng et al. (2024); Kwon et al. (2023) to support the RL algorithm and rollouts. The rollout number in RL is 16. We set the inference parameters as follows: $temperature = 0.6$, $top_p = 0.95$. For the LRM, we use a repetition penalty of 1.1, while for the LLM, the repetition penalty is set to 1.0. In the RL, the temperature of rollout is 1.0 and $top_p = 1.0$.

We conduct all experiments using 32 nodes with 8 NVIDIA H20 (96GB).

---

[3]https://github.com/QwenLM/Qwen-Agent/

E.4   Prompts for Agent Trajectories Sampling

Traditional `ReAct` for LLMs

Prompts for `ReAct`

Answer the following questions as best you can.

Use the following format:

Question: the input question you must answer
Thought: you should always think about what to do
Action: the action to take, should be one of [{tool_names}]
Action Input: the input to the action, use JSON Schema with explicit parameters
Observation: the result of the action
... (this Thought/Action/Action Input/Observation can be repeated **many** times)
Thought: you should always think about what to do
Action: Final Answer: the final answer to the original input question

Execution Framework
1. Thinking phase
- **Mandatory components**:
(a). Evidence chain completeness assessment
(b). Tool selection rationale

2. Action Phase
- **Allowed tools**: Only use tools listed in '{tool_descs}' or can be `Final Answer`, which returns the answer and finishes the task.
You may only provide the 'Final Answer' when you can confidently confirm the answer.
You must also ensure that the 'Final Answer' is accurate and reliable.
To output the Final Answer, use the following template: Final Answer: [YOUR Final Answer]

3. Observation phase
- **Return information from the tool**: The result of the action, you can use the result to think about the next step.
You have access to the following tools:

{tool_descs}

Begin!

You are likely to use the given tools to gather information and then make the final answer.
Solve the following question using interleaving thought, action, and observation steps. You may take as many steps as necessary.
Question: {query}

Figure 6: Prompts for `ReAct` using LLMs.

Modified `ReAct` for LRMs

Answer the following questions as best you can.
Allowed tools: Only use tools listed in {tool_descs}' or can be Final Answer: . You must also ensure that the Final Answer is accurate and reliable.
You have access to the following tools:
{tool_descs}

Begin!
Output Format:
Action: the action to take, should be one of [{tool_names}] Action Input: the input to the action, use JSON Schema with explicit parameters, when the action is 'Final Answer', do not have Action Input, directly return the answer
You may take as many steps as necessary. Always use the tools to gather information before making a final answer.
When you want to make a tool call, please output complete "Action: " and "Action Input: " to make the tool call successful and then output "Observation: " to make the tool call successful.
Question: {query}

Figure 7: Prompts for `ReAct` using LRMs.

F Case Study

As shown in Figure 8, several sophisticated thinking patterns are demonstrated as below:

- **Step-by-step Decomposition** WebDancer breaks down a complex problem into smaller, manageable steps with "First ... Then ... Finally".

- **Hypothesis Testing** WebDancer proposes hypotheses and verifies their validity. It assumes that "Nemo" from Finding Nemo is the orange clownfish (Amphiprion ocellaris). It validates this assumption by consulting the USGS database for non-native occurrences.

- **Handling Information Gaps** When certain information is missing, WebDancer takes action to fill the gap actively.

- **Iterative Reflection** After failing to find the ZIP code in the USGS database, WebDancer realizes it needs to search for the specific address separately. Through a second search, it successfully retrieves the ZIP code. This demonstrates the Agent's flexibility and adaptability when facing uncertainty.

Case Trajectory in GAIA

Question: I'm researching species that became invasive after people who kept them as pets released them. There's a certain species of fish that was popularized as a pet by being the main character of the movie Finding Nemo. According to the USGS, where was this fish found as a nonnative species, before the year 2020? I need the answer formatted as the five-digit zip codes of the places the species was found, separated by commas if there is more than one place.

Step 1

\<think\>
Okay, let's tackle this question step by step. The user wants to know the zip codes where the fish from Finding Nemo, which became invasive after being released by pet owners, was found as a

non-native species according to USGS before 2020. First, I remember that the main fish in Finding Nemo is Nemo, a clownfish. So the species in question is likely the orange clownfish, Amphiprion ocellaris. But I should confirm that. Maybe I should start by searching for c̈lownfish invasive species USGS̈to see if there's any info. Wait, but the user mentioned the USGS specifically. Let me check what USGS says about ...
</think>
<tool_call>
{"name": "search", "arguments": {"query": "USGS Amphiprion ocellaris nonnative locations before 2020"}}
</tool_call>
<tool_response>
A Google search for 'USGS Amphiprion ocellaris nonnative locations before 2020' found 9 results:
1. [clown anemonefish (Amphiprion ocellaris) - Species Profile](https://nas.er.usgs.gov/queries/FactSheet.aspx?speciesID=3243) Table 1. States with nonindigenous occurrences, the earliest and ...
2. [clown anemonefish (Amphiprion ocellaris)] (https://nas.er.usgs.gov/viewer/omap.aspx) Citation information: U.S. Geological Survey. [2025]. Nonindigenous Aquatic Species Database. Gainesville, Florida. Accessed [5/9/2025]. Contact us if you are...
...
</tool_response>

<think>
Okay, so I need to find the absolute difference in tens of thousands between the population of chinstrap penguins from the Wikipedia page as of the end of 2018 and the population from the 2020 Nature.com article. The user also wants us to assume two penguins per breeding pair. Let me start by going through the information provided.First, looking at the search results for the Wikipedia part. The sixth result mentions the Chinstrap penguin's conservation status, stating that in 2018, the IUCN estimated around 8 million specimens...
</think>
<tool_call>
{"name": "visit", "arguments": {"url": "https://nas.er.usgs.gov/queries/FactSheet.aspx?speciesID =3243", "goal": "Identify states with nonindigenous occurrences of Amphiprion ocellaris before 2020"}}
</tool_call>
<tool_response>
{"evidence": "The provided webpage content includes a table listing nonindigenous occurrences of Amphiprion ocellaris. The table shows that one individual was collected in Florida in July 2018 at Fred Howard Park, Pinellas Co., which is before 2...
</tool_response>

<think>
Okay, let's tackle this question step by step. The user is asking about the zip codes where the clownfish (Amphiprion ocellaris), made famous by Finding Nemo, was found as a nonnative species in the US before 2020 according to USGS. First, I did a search using the provided tools. The initial search led me to the USGS Nonindigenous Aquatic Species database page for Amphiprion ocellaris. Visiting that page showed a table listing occurrences. The key point here is that the only entry before 2020 is in Florida, specifically Pinellas County, collected in July 2018 at Fred Howard Park.Now, the next step is converting that location into a zip code ...
</think>

<tool_call>

{"name": "visit", "arguments": {"url": "https://nas.er.usgs.gov/queries/collectioninfo.aspx? SpeciesID=3243", "goal": "Find specific collection locations with zip codes for Amphiprion ocellaris in the US before 2020"}}

</tool_call>

<tool_response>

"evidence": "1468741 FL Pinellas Gulf of America, Florida, Fred Howard Park 2018 03100207 Crystal-Pithlachascotee eradicated", "rational": "The webpage provides a collection record for Amphiprion ocellaris (clown anemonefish) in the United States, specifically in Pinellas County, Florida. The record indicates that a specimen was collected at Fred Howard Park in 2018. However, the webpage does not provide zip codes for the collection locations." ...

</tool_response>

<div align="center">**Step 4**</div>

<think>

So now I need to figure out the zip code for Fred Howard Park in Pinellas County, Florida. Since the USGS data doesn't provide zip codes, I might need to look up the location's address. Let me think if there's another way using the available tools. The allowed tools are search and visit. Maybe I can perform another search to find the zip code of Fred Howard Park. Wait, the user's instructions say to use the tools provided. Since I can't just know the zip code offhand, I should use the search tool again to find the zip code of Fred Howard Park, Pinellas County, Florida. Let me try that. So next action: search for "Fred Howard Park Pinellas County Florida zip code". But I need to structure it properly. Let me check the parameters for the search tool. The query can be anything, so I can input "What is the zip code of Fred Howard Park, Pinellas County, Florida?" After getting that info, I can then present the zip code ...

</think>

<tool_call>

{"name": "search", "arguments": {"query": "Fred Howard Park Pinellas County Florida zip code"}}

</tool_call>

<tool_response>

A Google search for 'Fred Howard Park Pinellas County Florida zip code' found 9 results:

1. [Fred Howard Park - Pinellas County](https://pinellas.gov/parks/fred-howard-park/) Fred Howard Park is 155 acres of park and beach located off the Gulf of Mexico in Tarpon Springs. There is a 1-mile causeway from the park to the white sandy ...

2. [Fred Howard Park, 1700 Sunset Dr, Tarpon Springs, FL 34689, US](https://www.mapquest.com/us/florida/fred-howard-park-4792017) ...

</tool_response>

<div align="center">**Step 5**</div>

<think>

The first link is the species profile for Amphiprion ocellaris, which lists states with nonindigenous occurrences. The summary from visiting that page showed that Florida had a recorded occurrence in 2018 at Fred Howard Park in Pinellas County. Next, I needed to confirm the exact location details. The second visit to the collection info page provided a record from 2018 in Pinellas County, Florida, specifically Fred Howard Park. However, the zip code wasn't listed there, ... , and the specific location's zip code is 34689. Since the question asks for zip codes, and this is the only location mentioned in the provided data, the answer should be 34689. I need to ensure there aren't other locations, but based on the search results, only Florida's Pinellas County was cited. Thus, the final answer is the zip code 34689.

</think>

\<answer>34689 **\<answer>** ✓

Figure 8: Case Study.