WEBTRAVELER: A Multi-Agent Framework for Continuous Exploration in Website Traversal

Anonymous ACL submission

Abstract

Large language models (LLMs) are increasingly used as web-facing agents for information-seeking tasks over live websites. Many such tasks require multi-step in-site navigation and cross-page evidence aggregation, a setting known as web traversal. Existing LLM-based web agents, including recent explore–critic architectures, typically store interaction history as an ever-growing unstructured text trace and mix high-level planning with low-level page actions. This makes long-horizon behavior brittle, obscures the source of evidence, and often leads to redundant browsing or premature stopping. To mitigate these issues, we propose WEBTRAVELER, a hierarchical multi-agent framework that separates high-level information-seeking planning from local web exploration and maintains a structured trajectory memory of decision-relevant signals. This design provides compact information interfaces for evidence sufficiency checking and webpage navigation, improving long-horizon stability and cross-page evidence aggregation. To obtain scalable supervision, we further introduce path-guided data synthesis, which uses executable navigation paths to collect trajectories, removes explicit path hints, and filters invalid samples through answer-consistency checks. Experiments show that WEBTRAVELER consistently outperforms previous methods across LLM scales, especially on tasks requiring deeper navigation, with further gains from path-guided supervision. Moreover, WEBTRAVELER also shows generalization to broader web navigation tasks. Code is available at https://anonymous.4open.science/r/WebTraveler_anonymous-EB18.

1 Introduction

Large language models (LLMs) are increasingly used as web-facing agents that acquire up-to-date information by interacting with live websites rather than relying solely on parametric knowledge (Yang et al., 2025c; Ning et al., 2025). A dominant paradigm is to answer questions via web search (Nakano et al., 2022; Jin et al., 2025; Li et al., 2025c) and retrieval-augmented generation (RAG) (Lewis et al., 2020), where the model consumes a small set of retrieved passages or snippets. While effective for factoids, this paradigm may break down when the required information is not well exposed to keyword retrieval or is distributed across multiple pages. In practice, authoritative content is frequently organized behind navigational structures such as category trees, index pages and multi-level documentation; crucial details may only be reachable by following a specific sequence of in-site links. Answering such questions therefore requires multi-step in-site navigation, cross-page evidence aggregation, and traceable provenance for the collected evidence.

These requirements have motivated the formulation of *web traversal* (Wu et al., 2025b), where an agent starts from an entry page and collects sufficient evidence within the target site to answer a question. The agent must follow in-site links, reason over the navigation structure, and decide both which page to visit next and when the collected evidence is sufficient, even when relevant content lies several clicks away or spans multiple pages. This makes web traversal a long-horizon decision-making problem under partial observability, where robust cross-page evidence aggregation is essential rather than an optional post-processing step.

Within this broader space, a line of work has studied LLM-based web agents for open-domain browsing and web navigation (He et al., 2024; Wu et al., 2025b). For web traversal in particular, WebWalker (Wu et al., 2025b) adopts a ReAct-style explore–critic architecture (Yao et al., 2023), where the explorer performs navigation, and the critic maintains textual memory and decides when to answer. However, existing methods often maintain an ever-growing unstructured trace and couple high-level planning with low-level page interactions in

a single context. This design rapidly exhausts the context window of LLMs, making sustained exploration difficult. Moreover, it dilutes critical evidence and impairs reliable decision-making. These limitations call for a new framework with explicit information-state factorization and compact context management to support reliable decision-making during sustained long-horizon exploration.

To address these limitations, we propose WEBTRAVELER, a multi-agent framework that decomposes in-site web traversal into hierarchical planning and exploration. For planning, it introduces a Task Planner to maintain the global evidence states, check evidence sufficiency, and propose sub-goals for exploration if further exploration is needed. In exploration, the Perceiver converts each page observation into a compact digest, newly relevant evidence, and candidate buttons, while the Explorer selects executable navigation actions conditioned on the current goal and trajectory memory. At each interaction step, WEBTRAVELER updates a structured trajectory memory consisting of page digests, candidate actions, executed actions, and reflections. This compact memory replaces raw-history accumulation with decision-relevant states to help the agent stay focused and sustain long-horizon navigation under finite context budgets.

Beyond the architecture, we introduce path-guided data synthesis to obtain scalable supervision for the exploration roles. The pipeline first uses website structural priors to identify target pages reachable through deterministic, executable in-site paths, and then constructs question–answer pairs grounded in these pages. During trajectory collection, these paths guide successful in-site exploration, after which oracle hints are removed to avoid supervision leakage. We then apply answer-consistency checks to filter the synthesized trajectories to produce role-specific supervision aligned with the traversal objective. Experimental results on WebWalkerQA (Wu et al., 2025b) show that WEBTRAVELER consistently outperforms prior methods across LLM scales, with the advantage becoming larger on deeper trajectories and under larger token budgets. In addition, path-guided supervision brings further gains. Supplementary results on WebArena-Lite (Zhou et al., 2024; Liu et al., 2025b) further suggest that WEBTRAVELER can generalize to task-completion-oriented interactive web environments.

Our contributions can be summarized as follows:

- We present WEBTRAVELER, a hierarchical multi-agent framework that addresses the scalability and stability challenges of continuous website traversal with explicit role separation and compact trajectory memory.
- We introduce path-guided data synthesis, which turns executable navigation paths into scalable supervision while avoiding oracle-hint leakage.
- Experiments show that WEBTRAVELER outperforms competitive baselines across LLM scales, gains from path-guided supervision, and shows transfer potential on WebArena-Lite.

2 Related Work

LLM-based web agents have been widely studied for open-domain browsing and search-augmented question answering (Yao et al., 2022; Deng et al., 2023; Zhou et al., 2024; Xue et al., 2025; Mialon et al., 2024; Wei et al., 2025a; Zhou et al., 2025b). While effective for browser-using and broad information access, these settings often operate on top-ranked pages or snippet-level evidence, and may miss authoritative content that is only reachable via deeper in-site navigation. Closer to our setting, WebWalker (Wu et al., 2025b) formalizes web traversal and proposes an explore–critic agent built on the ReAct paradigm (Yao et al., 2023), but it still records history as an unstructured growing trace and couples high-level decisions with low-level actions in a single context. In contrast, WEBTRAVELER explicitly separates planning from local exploration and maintains a structured trajectory memory to improve stability on deep, in-site trajectories. On the agent training data side, previous datasets provide answer-level supervision or limited human demonstrations in browser environments (Shi et al., 2017; Yang et al., 2018; Welbl et al., 2018; Kwiatkowski et al., 2019; Deng et al., 2023; Zheng et al., 2024; Pan et al., 2024). Some work focuses on synthetic tool-using data with strong teachers (Qin et al., 2023; Tang et al., 2023), yet the resulting data are not necessarily grounded in real website settings. Recent efforts further scale web agent supervision by synthesising executable multi-step search trajectories on the web (Liu et al., 2025a; Wu et al., 2025a; Li et al., 2025b,d; Song et al., 2025) or by generating interaction demonstrations from web tutorials with automatic execution and verification (Xu et al., 2025;

Pahuja et al., 2025). In contrast, our method anchors trajectory generation on deterministic paths and applies rewriting and consistency checks to produce natural, page-grounded trajectories.

3 Method

3.1 Problem Formulation

The web traversal task can be formalized as follows. Given a natural language query q and an initial webpage (*i.e.*, the entry page of a target website), a system interacts with the website for multiple steps, collects evidence, and produces a final answer y. At step t, the system receives an observation o_t and executes an action a_t that transitions the environment to a new webpage. Following the environment construction in (Wu et al., 2025b), the observation is defined as $o_t = (p_t, l_t)$, where p_t is the page content in markdown format and $l_t = \{button_i\}_{i=1}^{K}$ is a set of interactable elements extracted by parsing the HTML. Each $button_i$ points to a target URL, and a_t selects one button to navigate.

Given this environment setting, previous agents are typically instantiated as a monolithic policy over the raw interaction history:

$$a_t \sim \pi_\theta(\cdot \mid q,\ o_t,\ \mathcal{H}_{t-1}^{\text{raw}}), \tag{1}$$

where $\mathcal{H}_{t-1}^{\text{raw}} = \langle o_0, a_1, \ldots, o_{t-1}, a_{t-1} \rangle$ denotes the raw trajectory. π_θ is a policy model (*e.g.*, an LLM). However, this paradigm couples planning, page understanding, and action decision within a single policy, imposing a heavy reasoning burden. Moreover, webpages often occupy thousands of tokens. Accumulating them linearly rapidly expands the context, making relevant evidence harder to retrieve and use reliably (Liu et al., 2024) and hindering sustained long-horizon exploration.

3.2 WebTraveler

Motivated by these limitations, we propose WEBTRAVELER, a hierarchical multi-agent framework. Inspired by human information seeking, where high-level intents guide low-level exploration actions, WEBTRAVELER organizes traversal into a planning level and an execution level. At the planning level, a Task Planner maintains the accumulated evidence and performs a sufficiency check to either terminate with a final answer or generate the next information-seeking sub-goals. At the execution level, a Web Exploration Loop phase executes the sub-goal through webpage interactions. Within this loop, an Explorer navigates and reflects on outcomes, while a Perceiver extracts goal-relevant evidence and compresses each observation into compact, decision-relevant signals. The extracted evidence is aggregated into a shared global evidence memory and fed back to the planner to perform the evidence sufficiency check. In parallel, the compressed interaction signals are organized in a structured trajectory memory, which replaces the ever-growing raw history and supports stable continuous exploration.

3.2.1 Global Planning

In web traversal, agents often suffer from aimless browsing or premature termination due to the lack of global directives. To address this, we introduce a Task Planner that maintains a global exploration state over the collected evidence. Specifically, acting as a plan-level controller, it focuses on the semantic gap between the user query q and the accumulated evidence $\mathcal{E}$. At the k-th planning round, the Task Planner π_{plan} performs a sufficiency check and outputs the current information goal g_k and task status σ_k:

$$g_k, \sigma_k \sim \pi_{\text{plan}}(q, g_{k-1}, \mathcal{E}). \tag{2}$$

$\sigma_k \in \{\text{COMP}, \text{CONT}\}$ denotes the decision status. If $\sigma_k = \text{COMP}$, the planner judges that $\mathcal{E}$ is sufficient to answer q and triggers final answer generation. Otherwise, it identifies the missing evidence and sets g_k as an information goal for the subsequent exploration loop.

3.2.2 The Web Exploration Loop

Given an information goal g_k from the Task Planner, the Web Exploration Loop executes it through repeated webpage interactions. Initially, the Explorer executes a fixed opening action to obtain the root observation o_1, and the Perceiver then initializes the root-page state $(d_1, \mathcal{C}_1, e_1)$. Within the loop, the Explorer decides navigation actions and performs reflection, while the Perceiver extracts evidence and decision-relevant signals from the resulting webpage for memory updates.

Execution & Reflection (Explorer). The Explorer is responsible for deciding navigation actions to fulfill the current sub-goal. However, web exploration is error-prone, as failed interactions can mislead subsequent decisions and accumulate over steps. To mitigate this, we further equip the Explorer with step-wise reflection on the previous interaction and adapt the next action accordingly.

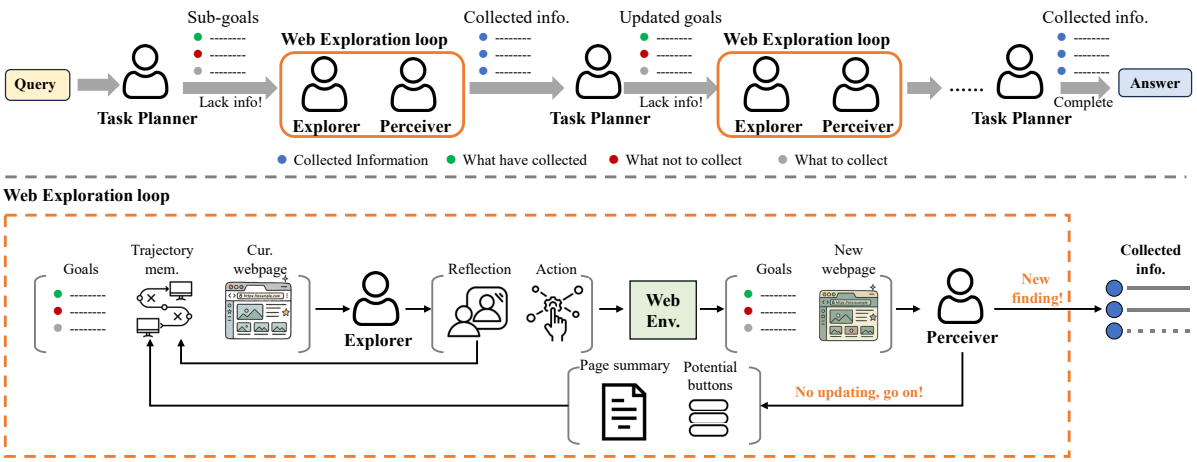

Figure 1: Overview of WEBTRAVELER. The Task Planner maintains global progress via evidence sufficiency checking and dispatches sub-goals. The Web Exploration Loop executes sub-goals with step-wise compression: the Explorer produces navigation actions together with lightweight reflections conditioned on the sub-goal and structured trajectory memory, while the Perceiver compresses page observations, selects potential interactive buttons, and extracts relevant evidence. By encoding every interaction into structured trajectory memory, WEBTRAVELER alleviates context growth and enables robust, continuous web traversal over long horizons.

Specifically, at step t, the Explorer performs reflection and decision-making by integrating the current observation o_t, latest transition $\tau_t = (o_{t-1}, a_{t-1}, o_t)$, the candidate buttons $\mathcal{C}_t$, and the compressed trajectory context $\mathcal{H}_t^{\text{ctx}}$:

$$a_t, r_t \sim \pi_{\text{exp}}(\cdot \mid q, g_k, \tau_t, \mathcal{C}_t, \mathcal{H}_t^{\text{ctx}}). \quad (3)$$

Here, $\mathcal{H}_t^{\text{ctx}} = \emptyset$ for $t = 1$ and $\mathcal{H}_{t-2}$ otherwise. The reflection r_t assesses whether the previous action succeeded and advanced the sub-goal g_k, while a_t is executed to continue exploration. The latest transition supports the Explorer for fine-grained reflection without relying on the full raw history.

Perception & Extraction (Perceiver). After executing a_t, the environment returns the next webpage observation o_{t+1}. Guided by the current information goal g_k, the Perceiver transforms the raw observation into compact semantic signals, including goal-relevant evidence, a page digest, and filtered candidate actions. Formally, the Perceiver ϕ_{per} produces three outputs:

$$(d_{t+1}, \mathcal{C}_{t+1}, e_{t+1}) = \phi_{\text{per}}(q, g_k, \mathcal{E}, o_{t+1}), \quad (4)$$

where k indexes planning rounds and t indexes environment interaction steps. Here, d_{t+1} is a compact digest of the newly reached page, $\mathcal{C}_{t+1}$ denotes the filtered candidate buttons for reducing the action space, and e_{t+1} is the evidence relevant to g_k. If $e_{t+1} \neq \emptyset$, we immediately add it to the global evidence memory (i.e., $\mathcal{E} \leftarrow \mathcal{E} \cup \{e_{t+1}\}$) and return it to the Task Planner for sufficiency check. Otherwise, the web exploration will continue.

Trajectory Memory. To support continual exploration under a finite context window, we maintain a structured trajectory memory that stores compact, decision-relevant signals instead of full page observations. At each step t, the memory is updated with the Perceiver's page digest, filtered candidate actions, the Explorer's selected action, and its step-wise reflection:

$$\mathcal{H}_t = \mathcal{H}_{t-1} \cup \{(a_t, d_{t+1}, \mathcal{C}_{t+1}, r_t)\}. \quad (5)$$

Each memory item records a completed transition: a_t is the action executed from o_t, while $(d_{t+1}, \mathcal{C}_{t+1})$ summarizes the resulting page o_{t+1}. This design replaces raw observations with compact semantic representations, enabling sustained exploration under a limited context window. By preserving these decision-relevant signals, the memory helps the Explorer avoid stagnation and repeated ineffective interactions in subsequent steps. (See Appendix A.2 for a specific example).

3.3 Path-guided Data Synthesis

WEBTRAVELER aims to stabilize long-horizon website exploration and cross-page evidence aggregation. However, general-purpose LLMs are not pretrained to align with the structural constraints of specific websites, limiting their ability to perform reliable web interactions. Meanwhile, high-quality traversal supervision is difficult to obtain: manual

Model	Framework	s-easy	s-med	s-hard	m-easy	m-med	m-hard	All
Qwen2.5-7B	ReAct	**42.86**	18.31	6.06	25.00	11.11	3.33	16.67
	WebWalker	39.29	**22.54**	**9.09**	8.33	**12.96**	3.33	16.67
	WEBTRAVELER	39.29	19.72	6.06	**33.33**	11.11	**10.00**	**18.33**
Qwen2.5-32B	ReAct	53.57	29.58	**15.15**	29.17	**16.67**	10.00	25.00
	WebWalker	50.00	33.80	12.12	25.00	14.81	**13.33**	25.00
	WEBTRAVELER	**64.29**	**40.85**	**15.15**	**33.33**	**16.67**	**13.33**	**30.42**
Qwen3-8B	ReAct	46.43	26.76	12.12	20.83	18.52	3.33	21.67
	WebWalker	35.71	23.94	9.09	12.50	11.11	**6.67**	17.08
	WEBTRAVELER	**53.57**	**29.58**	**15.15**	**41.67**	**24.07**	**6.67**	**27.50**
Qwen3-32B	ReAct	57.14	36.62	9.09	16.67	18.52	13.33	26.25
	WebWalker	**67.86**	35.21	**15.15**	20.83	9.26	10.00	25.83
	WEBTRAVELER	64.29	**39.44**	**15.15**	**29.17**	**31.48**	**23.33**	**34.16**

Table 1: Task success rate (%) on the WebWalkerQA benchmark without fine-tuning. "s-" and "m-" denote single-page and multi-page, respectively.

annotation is costly, while unconstrained synthesis may produce non-executable actions, invalid transitions, or answers unsupported by visited pages. We therefore propose path-guided data synthesis, which uses deterministic, executable navigation paths to provide guidance for generating reliable interaction agent trajectories.

Specifically, we first leverage website structural priors to mine target pages that are reachable through deterministic, executable in-site paths, and construct question–answer tasks grounded in the corresponding page content. Then, we use these question-answer tasks to drive the agent to collect interaction trajectories, while using the executable paths as step-wise guidance to provide structural references for page understanding and action decision-making. This design improves the success rate of reaching evidence pages in long-horizon exploration while ensuring that the collected trajectories remain reasonable. Finally, we remove explicit path hints and oracle traces through rewriting to avoid supervision leakage and apply answer-consistency checks to filter failed samples. The remaining trajectories reflect natural, page-grounded exploration and therefore provide high-quality supervision for model training. More details are provided in Appendix A.3.

4 Experiments

4.1 Experimental Settings.

Datasets. We evaluate WEBTRAVELER on WebWalkerQA (Wu et al., 2025b), a benchmark designed to measure web traversal ability by requiring an agent to navigate within a target website to answer a query. WebWalkerQA contains both single-page and multi-page tasks, and groups instances into easy, medium, and hard levels according to page-transition depth. In practice, however, WebWalkerQA is subject to website drift: some annotated answer pages are no longer accessible and some recorded navigation paths have changed, which can make individual instances non-executable or substantially alter their difficulty. Running evaluation naively on the full test set therefore mixes well-defined and ill-defined instances and undermines fair comparison between systems. To obtain a stable and reproducible evaluation protocol, we construct a curated subset of 240 instances by manually auditing the original test set and retaining only those whose answer pages are accessible and whose annotated paths still lead to the correct evidence pages. More details are provided in Appendix B.1.

Implementation Details. For evaluation, all methods are limited to at most 15 interaction steps and are executed under the same network environment to ensure consistent conditions. Following prior work (Wu et al., 2025b), we use task success rate as the primary metric and report results for each task category as well as overall. To ensure a fair comparison, we also adopt the LLM-as-a-Judge protocol to assess agent answers (Li et al., 2025a). For training data synthesis, to promote generality and avoid information leakage, we construct

Model	Easy	Medium	Hard	All
Qwen2.5-7B	36.54	16.00	7.94	18.33
Qwen2.5-7B*	**42.31**	**23.20**	**12.70**	**24.58**
Qwen3-8B	48.08	27.20	11.11	27.50
Qwen3-8B*	**59.62**	**32.00**	**19.05**	**34.58**
Qwen3-32B	48.67	36.00	19.05	34.16

Table 2: Results after fine-tuning on the synthesized data. "$*$" indicates fine-tuning.

Roles	Easy	Medium	Hard	All
Plan. + Per. + Exp.	**48.08**	**27.20**	**11.11**	**27.50**
Plan. + (Per./Exp.)	42.31	23.20	7.94	23.33
Per. + Exp.	42.31	25.60	6.35	24.17
All roles merged	44.20	22.40	6.35	22.92

Table 3: Architectural ablations using Qwen3-8B. Plan., Per., and Exp. denote the Planner, Perceiver, and Explorer, respectively. All variants keep the same memory and reflection mechanisms.

data only from publicly accessible websites with no overlap with WebWalkerQA. Additional implementation details are provided in Appendix B.2.

Baselines. We compare against two representative methods, ReAct (Yao et al., 2023) and WebWalker (Wu et al., 2025b), instantiated with backbone LLMs of varying scales, including Qwen2.5-7B, Qwen2.5-32B (Yang et al., 2025b), Qwen3-8B and Qwen3-32B (Yang et al., 2025a). ReAct is a general framework that interleaves reasoning and actions for multi-step web interaction, while WebWalker specialises in web traversal with an Explorer for navigation and evidence collection and a Critic that decides whether to continue exploring.

4.2 Main Results

In-context Learning Performance. Table 1 reports the results on the WebWalkerQA benchmark. Across all evaluated LLMs, the proposed WEBTRAVELER consistently achieves the best overall performance among the compared frameworks, indicating strong robustness to base models. The gains are more pronounced on stronger backbones. In particular, with Qwen3-32B, WEBTRAVELER achieves an overall success rate of 34.16%, outperforming ReAct (26.25%) and WebWalker (25.83%). The improvement is especially clear on multi-page tasks, where cross-page evidence aggregation is required. For example, under Qwen3-8B, WEBTRAVELER outperforms or matches both baselines on multi-page tasks across all three difficulty levels, suggesting that the proposed method is beneficial for sustained exploration.

Effect of Path-guided Synthesized Data. Table 2 reports the results after fine-tuning on trajectories synthesized by our proposed path-guided data synthesis method. The synthesized data provide consistent gains across base models. On Qwen2.5-7B, fine-tuning increases the overall success rate from 18.33% to 24.58% (+6.25%). On Qwen3-8B, it improves from 27.50% to 34.58% (+7.08%). Notably, performance on hard tasks increases markedly for both models (7.94%→12.70% and 11.11%→19.05%), indicating that the synthesized trajectories are particularly effective at strengthening long-horizon exploration and reasoning. Moreover, the fine-tuned Qwen3-8B reaches performance comparable to zero-shot Qwen3-32B, suggesting that path-guided supervision can partially narrow the gap to larger models under constrained model scale.

4.3 Ablation Studies

Architectural Ablations. Table 3 reports architectural ablations using Qwen3-8B as the base model. To isolate the effect of role decomposition, we compare the full three-role design with controlled variants that merge or remove roles. These merged-role variants preserve the same compressed memory and reflection interface while removing explicit role boundaries. As shown in the table, merging the Perceiver and Explorer reduces the overall success rate from 27.50% to 23.33%, and merging all roles into a single agent further reduces it to 22.92%. Using only the Perceiver and Explorer also reduces the overall success rate to 24.17%, highlighting the importance of the planner role. Overall, explicit role factorization and global evidence planning are both important for stable long-horizon traversal.

Component-wise Fine-tuning Ablations. Table 4 reports the results of fine-tuning different components of WEBTRAVELER using Qwen3-8B as the base model. Fine-tuning only the Perceiver improves the overall success rate from 27.50% to 29.17%, suggesting that better page abstraction and evidence extraction benefit long-horizon traversal. In contrast, fine-tuning only the Explorer decreases the overall success rate to 22.92%. This likely results from an interface mismatch: the

Method	Easy	Medium	Hard	All
W/o tuning	48.08	27.20	11.11	27.50
Only Perceiver	40.38	30.40	12.70	29.17
Only Explorer	40.38	22.40	9.52	22.92
Per. + Exp.	**59.62**	**32.00**	**19.05**	**34.58**

Table 4: Results of training different components (Qwen3-8B as the base model).

	Model	Easy	Medium	Hard	All
Evidence	Qwen3	88.69	74.63	77.38	80.23
	Qwen3-ft	89.64	82.83	82.76	85.08
Button	Qwen3	75.21	63.17	55.99	64.79
	Qwen3-ft	80.55	69.64	58.69	69.63

Table 5: Compression coverage diagnostics (using Qwen3-8B as the base model).

Explorer conditions on Perceiver-generated summaries, candidate buttons, and trajectory memory, but the Perceiver remains frozen. Jointly fine-tuning the Perceiver and Explorer achieves the best performance, improving the overall success rate to 34.58%. These results demonstrate that jointly training the interacting roles yields the best empirical configuration.

5 Discussions

Results Analysis by Exploration Horizon. To examine long-horizon exploration capability, we group WebWalkerQA tasks by the depth of the optimal access path to the answer page and report the results in Figure 2. Here, depth denotes the number of page transitions along the annotated optimal path, and we stratify tasks into three groups: depth 1–2, 3–4, and 5–6. On shallow tasks that require limited navigation and minimal global planning, all methods perform comparably, with WEBTRAVELER retaining a modest advantage. For tasks requiring deeper access paths, the advantage of WEBTRAVELER becomes more pronounced, with the largest margin observed on depth 5–6 tasks. The margin also grows when scaling the backbone from 8B to 32B. These results suggest that WEBTRAVELER better supports stable long-horizon exploration and can more effectively leverage stronger LLMs on deeper traversal tasks.

Efficiency Analysis under Token Budgets. Exploration efficiency is critical for effective web traversal. However, comparing task latency is confounded by exploration horizon: agents that stop early may appear faster, while agents that continue exploring and solve additional instances can be underestimated. We therefore use the total number of tokens consumed per task as the budget metric and compare how many instances each method solves under different token limits, measuring budget utilization efficiency. As shown in Figure 3, ReAct attains higher coverage under small token budgets, mainly due to its lower startup overhead from the single-role design. However, as the budget increases, WEBTRAVELER continues to solve more tasks, while ReAct and WebWalker saturate earlier. This suggests that simply allocating more tokens does not guarantee higher task coverage, and that the agent architecture itself can become a bottleneck for performance. In contrast, structured trajectory memory and explicit evidence planning allow WEBTRAVELER to sustain more effective long-horizon navigation and evidence collection, leading to a widening advantage as the budget increases. The fine-tuned WEBTRAVELER (8B*) further improves this trend, with smoother and stronger gains as the budget increases, indicating that path-guided synthesized data improves the exploration efficiency.

Compression Coverage Diagnostics. Since WEBTRAVELER relies on Perceiver compression to maintain a compact trajectory memory, we further examine whether the Perceiver compression discards decision-critical information. Using annotated answers and golden paths, we apply Qwen3-Max (Yang et al., 2025a) to evaluate two coverage diagnostics at each eligible step: *evidence coverage*, which measures whether task- or answer-relevant evidence on the current page is included in the Perceiver output, and *button coverage*, which measures whether the golden-path button is retained in `potential buttons`. We compute coverage at the task level and report the results in Table 5. As shown in the table, evidence coverage reaches 80.23% overall, suggesting that most task-relevant evidence is preserved. Button coverage is relatively lower, especially on hard tasks, where decision-making is inherently more challenging and key buttons are more likely to be ignored. Moreover, fine-tuning improves evidence coverage from 80.23% to 85.08% and button coverage from 64.79% to 69.63%, indicating that path-guided fine-tuning can mitigate information loss and improve

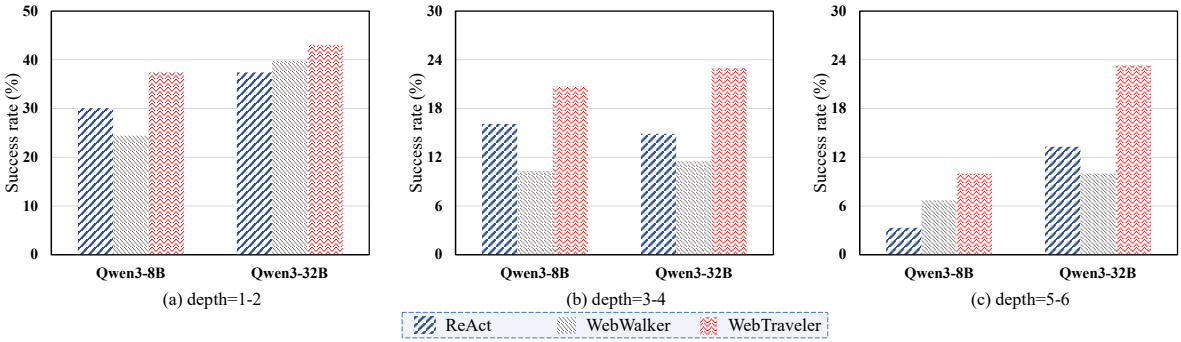

Figure 2: Results across tasks with different answer-page depths.

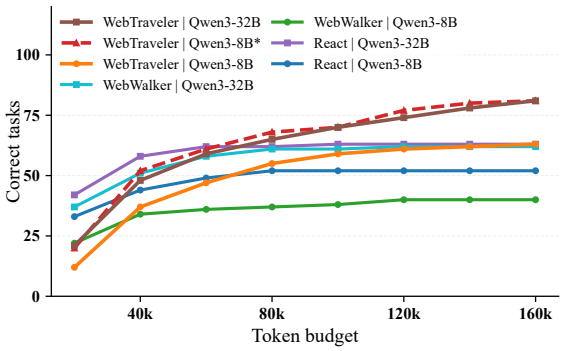

Figure 3: Number of correctly solved tasks under varying token budgets.

Method	Red.	Git.	CMS	Map	Shop.	AVG
Prompting Method						
WR-base	**5.3**	**10.0**	5.7	**13.3**	8.9	8.5
WEBTRAVELER	**5.3**	6.7	**11.4**	**13.3**	**13.3**	**10.3**
Fine-tuned						
WR-BC	**36.8**	6.7	**20.0**	**33.3**	17.8	20.6
WEBTRAVELER	21.1	**23.3**	**20.0**	30.0	**22.2**	**22.4**

Table 6: Results on WebArena-Lite (Llama3.1-8B-Instruct as the base model).

the reliability of the Perceiver interface.

Generalization to Broader Web Environments. To examine whether WEBTRAVELER generalizes beyond evidence-seeking traversal, we further evaluate it on WebArena-Lite (Zhou et al., 2024; Liu et al., 2025b), a task-completion benchmark with interactive web tasks across Reddit, GitLab, CMS, Map, and Shopping websites. For fair comparison, we conduct experiments using Llama3.1-8B-Instruct (Grattafiori et al., 2024) as the base model, retain the multi-role agent structure and trajectory memory mechanism of WEBTRAVELER, while aligning the tool interface and observation format with WebAgent-R1 (Wei et al., 2025b).

We first compare prompting methods without training. As shown in Table 6, WEBTRAVELER achieves the best average success rate under the prompting-only setting, outperforming the non-trained baselines of WebAgent-R1 (WR-base). These results suggest that the proposed agent framework WEBTRAVELER is also effective in interactive task-completion environments, demonstrating its generality beyond the WebWalkerQA setting.

We then conduct fine-tuning experiments to examine whether the proposed path-guided data synthesis can transfer to broader interactive web environments. For a fair comparison, we use the same step-level behavior cloning (BC) demonstrations used for SFT in WebAgent-R1, and treat their executable action traces as path-like guidance for our data synthesis. Table 6 reports the results. Compared with its prompting-only counterpart, the fine-tuned WEBTRAVELER improves from 10.3% to 22.4%, with gains across all five domains. It also outperforms WR-BC, the behavior-cloning trained WebAgent-R1 baseline. This indicates that the proposed path-guided synthesis method has transfer potential as a general data synthesis paradigm for broader interactive web environments.

6 Conclusion

We present WEBTRAVELER, a hierarchical multi-agent framework for stable long-horizon website traversal. The framework separates planning from local exploration and uses structured trajectory memory to support compact, evidence-aware decision making. To provide scalable supervision, we introduce path-guided data synthesis, which converts executable website paths into page-grounded training trajectories without oracle-hint leakage. Experiments demonstrate consistent gains across LLM scales, especially on deeper traversal tasks, and further show that WEBTRAVELER can transfer to broader interactive web-agent settings.

Limitations

Manually Constrained Agent Actions and Observations. Although our experiments cover both WebWalkerQA and WebArena-Lite, the agent actions and observations are still constrained by benchmark-specific tool interfaces rather than the full space of real-world browser interactions. As foundation models become more powerful, future work could move toward more complete browser action spaces and richer observations, enabling agents to operate with fewer manually imposed interface constraints.

Potential Information Loss. To control context growth, our method compresses each page into a compact semantic summary. Although we provide coverage diagnostics to analyze whether the compression preserves decision-critical evidence and buttons, this design may still discard fine-grained evidence or mistakenly prune critical actions. Future improvements in webpage summarization and perception could further improve the quality of compressed memory while reducing unnecessary context and invocation overhead.

Higher Startup Overhead. WEBTRAVELER consists of multiple roles which introduce additional invocation costs and engineering complexity. For shallow questions that require only a few interaction steps, this startup overhead can be relatively more pronounced.

References

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. In *Advances in Neural Inf. Process. Syst.*, pages 28091–28114.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. Webvoyager: Building an end-to-end web agent with large multimodal models. In *Proc. Conf. Association for Computational Linguistics*, pages 6864–6890.

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

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur 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. 2019. Natural questions: A benchmark for question answering research. *Trans. Assoc. Comput. Linguist.*, 7:452–466.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Inf. Process. Syst.*, 33:9459–9474.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, Kai Shu, Lu Cheng, and Huan Liu. 2025a. From generation to judgment: Opportunities and challenges of llm-as-a-judge. In *Proc. Conf. Empirical Methods in Natural Language Processing*, pages 2757–2791.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, Weizhou Shen, Junkai Zhang, Dingchu Zhang, Xixi Wu, Yong Jiang, Ming Yan, Pengjun Xie, Fei Huang, and Jingren Zhou. 2025b. Websailor: Navigating super-human reasoning for web agent. https://arxiv.org/abs/2507.02592. ArXiv:2507.02592.

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

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

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

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Trans. Assoc. Comput. Linguist.*, 12:157–173.

Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Yifan Xu, Xixuan Song, Shudan Zhang, Hanyu Lai, Xinyi Liu, Hanlin Zhao, Jiadai Sun, Xinyue Yang, Yu Yang, Zehan Qi, Shuntian Yao, Xueqiao Sun, Siyi Cheng,

Qinkai Zheng, Hao Yu, and 11 others. 2025b. Visualagentbench: Towards large multimodal models as visual foundation agents. In *Proc. Int. Conf. Learn. Representations*.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. Gaia: a benchmark for general ai assistants. In *Proc. Int. Conf. Learn. Representations*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. Webgpt: Browser-assisted question-answering with human feedback. *Preprint*, arXiv:2112.09332.

Liangbo Ning, Ziran Liang, Zhuohang Jiang, Haohao Qu, Yujuan Ding, Wenqi Fan, Xiao yong Wei, Shanru Lin, Hui Liu, Philip S. Yu, and Qing Li. 2025. A survey of webagents: Towards next-generation ai agents for web automation with large foundation models. In *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*, pages 6140–6150.

OpenAI. 2025. Introducing gpt-5.

Vardaan Pahuja, Yadong Lu, Corby Rosset, Boyu Gou, Arindam Mitra, Spencer Whitehead, Yu Su, and Ahmed Hassan Awadallah. 2025. Explorer: Scaling exploration-driven web trajectory synthesis for multimodal web agents. In *Findings Assoc. Comput. Linguist.*, pages 6300–6323.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, and Zhengyang Wu. 2024. Webcanvas: Benchmarking web agents in online environments. *Preprint*, arXiv:2406.12373.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *Preprint*, arXiv:2307.16789.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. 2017. World of bits: An open-domain platform for web-based agents. In *Proc. Int. Conf. Mach. Learn.*, pages 3135–3144. PMLR.

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

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. 2023. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *Preprint*, arXiv:2306.05301.

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

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. 2025b. WebAgent-r1: Training web agents via end-to-end multi-turn reinforcement learning. In *Proc. Conf. Empirical Methods in Natural Language Processing*, pages 7909–7928.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Trans. Assoc. Comput. Linguist.*, 6:287–302.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Gang Fu, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. 2025a. Webdancer: Towards autonomous information seeking agency.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. 2025b. WebWalker: Benchmarking LLMs in web traversal. In *Proc. Conf. Association for Computational Linguistics*.

Yiheng Xu, Dunjie Lu, Zhennan Shen, Junli Wang, Zekun Wang, Yuchen Mao, Caiming Xiong, and Tao Yu. 2025. Agenttrek: Agent trajectory synthesis via guiding replay with web tutorials. In *Proc. Int. Conf. Learn. Representations*.

Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. 2025. An illusion of progress? assessing the current state of web agents. In *Proc. Conf. Language Modeling*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Qwen: An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2025b. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Yingxuan Yang, Mulei Ma, Yuxuan Huang, Huacan Chai, Chenyu Gong, Haoran Geng, Yuanjian Zhou, Ying Wen, Meng Fang, Muhao Chen, Shangding Gu, Ming Jin, Costas Spanos, Yang Yang, Pieter Abbeel, Dawn Song, Weinan Zhang, and Jun Wang. 2025c. Agentic web: Weaving the next web with ai agents. *Preprint*, arXiv:2507.21206.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proc. Conf. Empirical Methods in Natural Language Processing*, pages 2369–2380.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Inf. Process. Syst.*, 35:20744–20757.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *Proc. Int. Conf. Learn. Representations*.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024. Gpt-4v(ision) is a generalist web agent, if grounded. In *Proc. Int. Conf. Mach. Learn.*

Junting Zhou, Wang Li, Yiyan Liao, Nengyuan Zhang, Tingjia Miao, Zhihui Qi, Yuhan Wu, and Tong Yang. 2025a. Scholarsearch: Benchmarking scholar searching ability of llms. *Preprint*, arXiv:2506.13784.

Peilin Zhou, Bruce Leon, Xiang Ying, Can Zhang, Yifan Shao, Qichen Ye, Dading Chong, Zhiling Jin, Chenxuan Xie, Meng Cao, Yuxin Gu, Sixin Hong, Jing Ren, Jian Chen, Chao Liu, and Yining Hua. 2025b. Browsecomp-zh: Benchmarking web browsing ability of large language models in chinese. *Preprint*, arXiv:2504.19314.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. Webarena: A realistic web environment for building autonomous agents. In *Proc. Int. Conf. Learn. Representations*.

A Method Details

A.1 WEBTRAVELER Inference Procedure

Algorithm 1 shows the detailed inference procedure of WEBTRAVELER. The formulation in Section 3.1 describes web traversal in a general conditioned policy process. WEBTRAVELER instantiates this policy with a factorized multi-agent implementation. Instead of using one monolithic policy over the raw history, it replaces $a_t \sim \pi_\theta(\cdot \mid q, o_t, \mathcal{H}_{t-1}^{\text{raw}})$ with a staged decision process: $(g_k, \sigma_k) \sim \pi_{\text{plan}}(q, g_{k-1}, \mathcal{E})$, $(a_t, r_t) \sim \pi_{\text{exp}}(\cdot \mid q, g_k, o_t, o_{t-1}, a_{t-1}, \mathcal{C}_t, \mathcal{H}_t^{\text{ctx}})$, and $(d_{t+1}, \mathcal{C}_{t+1}, e_{t+1}) = \phi_{\text{per}}(q, g_k, \mathcal{E}, o_{t+1})$. The agents communicate through two explicit shared states, the evidence set $\mathcal{E}$ and the structured trajectory memory $\mathcal{H}$, which replace the raw history $\mathcal{H}^{\text{raw}}$ with compact decision-relevant states. If the Perceiver identifies new evidence e_{t+1}, $\mathcal{E}$ is updated and the Planner revises the goals or terminates the exploration if the evidence is sufficient.

Algorithm 1 WEBTRAVELER Inference Procedure

Require: query q, root webpage u_0, maximum policy-selected interaction steps T
Ensure: final answer y

Initialize accumulated evidence $\mathcal{E} \leftarrow \emptyset$
Initialize previous observation $o_0 \leftarrow \emptyset$
Initialize previous information goal $g_0 \leftarrow \emptyset$
Initialize planner revision index $k \leftarrow 1$
$(g_1, \sigma_1) \leftarrow \pi_{\text{plan}}(q, g_0, \mathcal{E})$

$a_0 \leftarrow \text{OPEN}(u_0)$, $r_0 \leftarrow \emptyset$
Execute fixed Explorer action a_0 and obtain root observation o_1
$(d_1, \mathcal{C}_1, e_1) \leftarrow \phi_{\text{per}}(q, g_1, \mathcal{E}, o_1)$
$\mathcal{H}_0 \leftarrow \{(a_0, d_1, \mathcal{C}_1, r_0)\}$
if $e_1 \neq \emptyset$ **then**
  $\mathcal{E} \leftarrow \mathcal{E} \cup \{e_1\}$
  $k \leftarrow k + 1$
  $(g_k, \sigma_k) \leftarrow \pi_{\text{plan}}(q, g_{k-1}, \mathcal{E})$
end if
for $t = 1$ to T **do**
  if $\sigma_k = \text{COMP}$ **then**
    break
  end if
  $\mathcal{H}_t^{\text{ctx}} \leftarrow \emptyset$ if $t = 1$, otherwise $\mathcal{H}_{t-2}$
  $(a_t, r_t) \leftarrow \pi_{\text{exp}}(q, g_k, o_t, o_{t-1}, a_{t-1}, \mathcal{C}_t, \mathcal{H}_t^{\text{ctx}})$
  Execute Explorer action a_t and obtain observation o_{t+1}
  $(d_{t+1}, \mathcal{C}_{t+1}, e_{t+1}) \leftarrow \phi_{\text{per}}(q, g_k, \mathcal{E}, o_{t+1})$
  $\mathcal{H}_t \leftarrow \mathcal{H}_{t-1} \cup \{(a_t, d_{t+1}, \mathcal{C}_{t+1}, r_t)\}$
  if $e_{t+1} \neq \emptyset$ **then**
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{e_{t+1}\}$
    $k \leftarrow k + 1$
    $(g_k, \sigma_k) \leftarrow \pi_{\text{plan}}(q, g_{k-1}, \mathcal{E})$
  end if
end for
$y \leftarrow \pi_{\text{ans}}(q, \mathcal{E})$
return y

A.2 Example of Structured Trajectory Memory

Figure 4 presents a representative memory item from the structured trajectory memory. Each item records the step index, executed action, a compact observation summary with candidate buttons, and a step-wise reflection. This structured representation preserves traversal-relevant signals while avoiding the context length growth associated with storing raw webpage observations.

Example of Structured Trajectory Memory

Step: 4

Policy:
Action: `visit_page`; Parameters: {`button`: "Executive Committee and Council Members"}

Observation:
The page lists the Institute of Navigation's governance structure, including roles such as President, Executive Vice President, Treasurer, and other committee members. Contact details and links to related sections are also provided.

Clickable button candidates:
1) Governance; 2) About ION; 3) Membership; 4) Contact ION; 5) News and Announcements.

Reflection:
The previous strategy efficiently narrowed the focus to executive leadership, but has not yet checked the contact page, which may contain direct contact information. The next step should prioritize "Contact ION" while avoiding revisiting already examined pages such as "Council" or "Governance".

Figure 4: Example of Structured Trajectory Memory. The query is "Who is the Executive Director of the Institute of Navigation?".

A.3 More Path-guided Data Synthesis Details

To make the path-guided data synthesis process clearer, we provide an illustration in Figure 5. Specifically, we first sample target pages from real websites and construct page-grounded QA tasks together with deterministic, executable in-site paths. The QA task serves as the exploration objective, while the executable path is used only during trajectory collection.

During collection, the path is provided as an auxiliary prompt-level guidance field rather than as part of the webpage observation. At each step, this field indicates the next structurally relevant direction, such as the next target link or page section to inspect. The agent still observes the normal page content and available clickable elements,

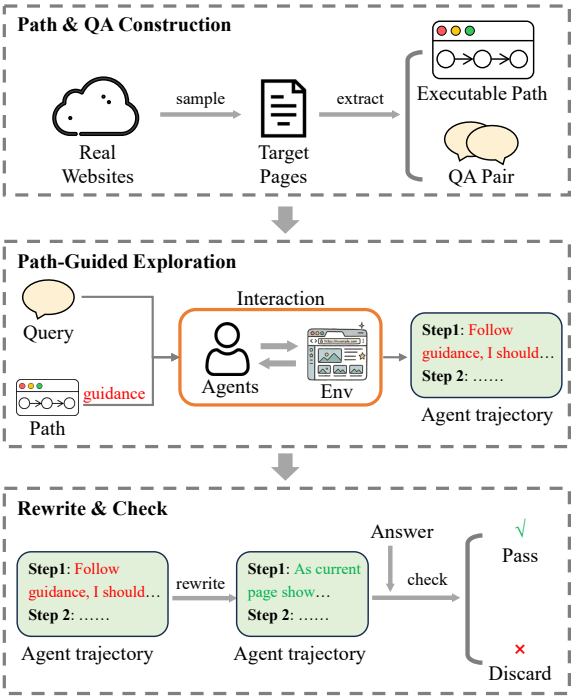

Figure 5: Overview of the path-guided data synthesis.

Category	#Tasks
single-easy	28
single-medium	71
single-hard	33
multi-easy	24
multi-medium	54
multi-hard	30
ALL	240

Table 7: statistics of the evaluation dataset.

Framework	Model	orig.	filt.
ReAct	Qwen2.5-7B	16.02	16.67
ReAct	Qwen2.5-32B	25.44	25.00
WebWalker	Qwen2.5-7B	19.85	16.67
WebWalker	Qwen2.5-32B	26.02	25.00

Table 8: Overall task success rate (%) on original WebWalkerQA (orig.) and the filtered subset (filt.).

and produces standard Perceiver and Explorer outputs. Thus, the guidance increases the probability of reaching the evidence page, but it is not included in the final training input.

After collecting the raw trajectory, we rewrite the agent outputs to remove explicit guidance signals and oracle traces. In particular, statements such as "following the guidance, I should click ..." are rewritten into observation-grounded rationales, e.g., "the current page contains a relevant link for the missing information." We then apply consistency checks before retaining a sample. A trajectory is kept only if its actions are executable, its final answer is supported by the visited pages, and the rewritten trajectory no longer contains explicit path hints.

B Experimental Details

B.1 Benchmark

We focus on website traversal rather than search-centric benchmarks such as ScholarSearch (Zhou et al., 2025a) and BrowseComp (Wei et al., 2025a). These benchmarks primarily evaluate an agent's ability to locate information from the open web, often relying on search engines, indexed snippets, or top-ranked result pages as entry points. While valuable for evaluating general information-seeking ability, they do not directly target the core challenges studied in this work: multi-step in-site navigation, executable action selection, and cross-page evidence aggregation within a specified target website.

In contrast, WebWalkerQA requires an agent to start from a given website entry page, follow in-site links, and collect sufficient evidence from the target site to answer the query. Therefore, we use WebWalkerQA as the primary benchmark for evaluating long-horizon website traversal ability.

WebWalkerQA is constructed over live, real-world websites. In our reproduction, we found that a non-trivial fraction of the original tasks has become unstable or ill-posed over time due to website drift and annotation inconsistencies. This instability adds evaluation noise that is largely orthogonal to agent capability, such as broken navigation paths, inaccessible evidence sources, or pages whose content has changed. To obtain a controlled and reproducible evaluation protocol, we manually audited the benchmark. For each task, we verified that the root URL and the annotated navigation path remain executable, and that the labeled answer page is reachable and contains sufficient evidence to support the reference answer (either explicitly stated or derivable via straightforward reasoning). Following this audit, we curate a stable subset of 240 tasks. Dataset statistics for the resulting subset are reported in Table 7.

To check whether the filtered subset remains representative of the original benchmark, we compare success rates reported for the full WebWalkerQA in the original paper with our reproduced

Category	Sites
General News	The Guardian / NPR / USA Today China News Service / Sina News / Jiemian News
Tech / Developer	The Verge / GitLab Docs / MDN Web Docs Open Source China (OSChina) / SegmentFault / Runoob Tutorial
Education / Universities	Harvard University / University of Cambridge / University of Oxford Tianjin University (CIC) / Shanghai Jiao Tong University (SEIEE) / SUSTech (CSE)
Healthcare / Medicine	NHS / Verywell Health / MedlinePlus People.cn Health / FamilyDoctor / Haodf (Good Doctor Online)
Finance / Business	Investopedia / Nasdaq / CNBC Securities Times / Yicai / 21st Century Business Herald
Culture / Arts	The Metropolitan Museum of Art / Tate / Victoria and Albert Museum CAFA Art Museum / Zhishifenzi / China Writer
E-commerce / Retail	Decathlon / MUJI (US) / Zappos OPPO Store / Haier / Xiaomi Store
Sports	Yahoo Sports / Sports Illustrated / CBS Sports CCTV Sports / Dongqiudi / NetEase Sports
Entertainment / Media	Metacritic / RogerEbert.com / The A.V. Club 1905 Movie Network / Mango TV / China Film Archive
Journals / Conferences	Nature / NeurIPS / SIGGRAPH Chinese Journal of Image and Graphics / Robot (SIA) / VALSE

Table 9: Public websites used for Path-Guided synthesis.

results on the 240-task subset. We consider two representative baselines, ReAct (Yao et al., 2023) and WebWalker (Wu et al., 2025b), instantiated with Qwen2.5-7B and Qwen2.5-32B as the base LLMs. As shown in Table 8, the relative performance trends are highly consistent between the full benchmark and the subset, suggesting that the subset largely preserves the benchmark's difficulty profile while removing unstable instances.

B.2 More Implementation Details

Evaluation. For evaluating webwalkerQA, to ensure a fair and consistent comparison, we evaluate all methods using the same automatic judge, GPT-5.1 (OpenAI, 2025). For each example, the judge takes the query, the expected answer, and the agent-generated answer as inputs, and outputs a binary score in $\{0, 1\}$ indicating whether the agent's answer is correct. The evaluation prompt is shown in Figure 6.

For evaluating WebArena-Lite, we follow the official WebArena evaluation protocol and use the benchmark-provided task-specific automatic evaluators. These evaluators assess task success through string matching, URL matching, and programmatic webpage-state/content checks, depending on the task configuration.

Training details. We construct the training data using the proposed path-guided synthesis over 60 publicly accessible and compliant websites, thereby minimizing potential risks in the collected data (e.g., personally identifying info or offensive content). For each website, we collect 40 single-page QA trajectories and 20 multi-page QA trajectories, preventing the dataset from being dominated by any single site or interaction pattern. The resulting training set contains 2,400 single-page QA pairs and 1,200 multi-page QA pairs, comprising 10,052/6,653 Perceiver messages and 7,553/5,332 Explorer messages for single/multi-page tasks, respectively. Notably, whereas WebWalkerQA restricts target websites to four domains (conference, organization, education, and game), we intentionally cover 10 domains. As a result, our website set is broader and more balanced in domain coverage, and is not tailored to any specific WebWalkerQA sites (The complete list is provided in Table 9). This reduces the risk of benchmark-specific overfitting and better demonstrates the generality and transferability of our synthesized data and training pipeline.

For training, both the Qwen2.5-7B, Qwen3-8B, and Llama3.1-8B-Instruct are trained using two NVIDIA H100 GPUs with two epochs. The maximum context length is set to 16K tokens.

Prompt for Judge

You are a world-class answer consistency verifier. I will provide three inputs: 1. query (the question) 2. expected_answer (the reference answer) 3. agent_answer (the model-generated answer)
Your task is to strictly judge whether agent_answer correctly answers the query according to expected_answer, and produce:
- Point-by-point analysis (coverage and consistency) - A clear conclusion - Final score (`1` = success; `0` = failure)

Principles:
- (General · Query-first-1) This task treats the query as the only source to determine the mandatory set of key points (Q-key points). expected_answer is only a reference for correct wording and granularity for calibration and alignment; you must not introduce new mandatory points beyond the query.
- (General · Query-first-2) Any background information or extra constraints described in the query do not have to be restated in agent_answer; do not give 0 merely because agent_answer does not repeat such background or constraints.

A. Key-point Coverage
- If agent_answer covers all key points in expected_answer (i.e., all major facts/information points), then output 1 (success). Different wording or extra information is allowed, but must not contradict expected_answer.
- If agent_answer omits, is vague about, or incorrectly answers any key point in expected_answer, then output 0 (failure).
- (Practical note) If agent_answer clearly and accurately covers the core facts (defined below) and has no conflict with expected_answer, you may output 1. For non-decisive modifiers (defined below), omission is allowed and should not be penalized.
- (Query alignment) Only information points that directly answer the query are treated as key points. Content in expected_answer that does not directly answer the query and does not change the conclusion is treated as supplementary; missing it should not be penalized.
- Key-point source order: First, extract the minimal necessary points from the query as Q-key points; then use expected_answer to calibrate their definition/granularity/pairing. Only when missing details would change the definition, unit, denominator, scope, or time granularity should they be upgraded into Q-key points.
- If expected_answer contains information beyond the query, default it to S-supplementary points; missing it is not penalized. Only if agent_answer explicitly contradicts an S-point should you output 0.
- Mandatory labeling (Q/S): Before judging, label points from expected_answer into:
Q-key points: facts necessary to directly answer the query, or details whose absence would change definition/unit/denominator/scope/time granularity.
S-supplementary points: all other background/expansion details, as well as all background constraints explicitly stated in the query (they are not core facts).
- Missing handling: scoring is based only on Q-key points. If agent_answer covers all Q-key points with no contradiction, output 1. Missing S-points does not reduce score; only explicit contradiction with any S-point leads to 0.

B. No Subjective Inference or Vague Coverage
- Do not treat "related/similar" information as "covered." Coverage requires agent_answer to explicitly provide an equivalent fact (or clear synonym) to expected_answer.
- If agent_answer uses uncertainty/speculation words (e.g., "maybe," "probably," "possibly," "seems," etc.) regarding any expected key point, treat it as not covered.
- (Definition invariance) If missing or altering a detail would materially change the definition/meaning (e.g., population scope, geography, unit/denominator, statistical definition, time granularity), that detail is automatically a key point. Missing it or contradicting expected_answer yields 0.

- (Background constraints) All background information explicitly stated in the query is not treated as a core fact; even if expected_answer includes it, agent_answer is not required to restate it.
- expected_answer must not expand the query scope: never treat extra details in expected_answer as new Q-key points unless missing them would change the definition and thus must be upgraded by the rules above.

C. Date and Pairing Consistency
- If expected_answer specifies date–event (or date–point) pairings, agent_answer must match the same date and event type one by one and explicitly.
- Mentioning other content in the same period without the correct event/announcement type is treated as not covered.

D. Extra Information Rules
- Extra details beyond expected_answer are allowed as long as they do not contradict expected_answer.
- Any contradiction with expected_answer leads to 0.

E. Format and Output Requirements
- You must output a raw JSON object (not a string; do not wrap in code fences; no pre/-post explanatory text).
- JSON fields are fixed and must be complete:
"query"
"expected_answer"
"agent_answer"
"analysis": one paragraph in Chinese with point-by-point, checkable analysis. It must explicitly state whether each Q-key point is covered, and whether any S-point has contradictions.
"score": must be `0` or `1`.
- Do not output anything other than the JSON object.

Core Facts vs. Modifiers (General Definitions)
- Core facts (typically must be explicitly covered):
numbers/ratios/levels; precise time anchors (dates/intervals); main entities (policy/agency/target); action/event types (publish/implement/increase/-support/etc.); directional conclusions (up/down/achieved/not achieved).
- Modifiers (can be omitted unless they change the definition):
population scope restrictions, region/channel/scenario lists, taxonomies, methodology/measurement details, implementation details, background descriptions, adjectival qualifiers.
If omission/change would materially alter definition/unit/denominator/scope/time granularity, upgrade to a key point.

Judgment Procedure
1) Q/S labeling (Query → Expected): extract Q-key points from the query only, then calibrate with expected_answer; upgrade to Q only when necessary to preserve definition/unit/denominator/scope/time granularity.
2) Query alignment check: ensure all Q-key points directly answer the query; keep non-essential expected_answer details as S.
3) Point-by-point verification: check agent_answer explicitly and equivalently covers all Q-key points; uncertainty language does not count as coverage.
4) Supplement consistency: for S-points, only check contradictions; missing is not penalized.
5) Contradictions and vagueness: apply rules for conflicts and vague wording.
6) Conclusion and score: all Q covered with no contradictions → score = 1; otherwise score = 0.

Please strictly follow the query-first principles and the rules above to judge whether agent_answer correctly answers the query. You must output only a JSON object and nothing else.

Figure 6: Prompts for Evaluation.

C Prompts in WebTraveler

The prompts in WebTraveler are shown in Figure 7 (for Task Planner), Figure 8 (for Perceiver) and Figure 9 (for Explorer).

Prompt for Task Planner

You are a world-class Task Planner. You need to decompose the user's task into clear information-gathering subgoals, and at each update of information, determine whether the collected information is sufficient to complete the task.

Goal:
- Record information points that do not need to be collected (info_not_to_collect), as they have been verified or are irrelevant to the task.
- Identify key information points required to continue completing the task (info_to_collect).
- If the collected knowledge (directly or by logical implication) is sufficient to complete the task, set complete = true.

Input:
- Query: "<user's question>"
- Last Info_to_collect: "<previous information-gathering goals>"
- Last Info_not_to_collect: "<previously verified or irrelevant information points>"
- Current Collected_information: "<currently collected information>"

Output format:
<reasoning>
Briefly reason from the following aspects:
- To complete the task, which information has been verified and no longer needs to be collected, which is still missing, and how it can be gathered.
- Whether the currently collected information can answer the user's question.
</reasoning>
{
"info_not_to_collect": ["information items that have been verified or are irrelevant to the task"],
"info_to_collect": ["information items that still need to be collected (not steps; must include missing information required to complete the task; may include inferred conditions and synonym variants)"],
"complete": true or false, output true when the collected information can answer the user's question; otherwise output false. Important: If the user has not specified details, do not over-extend the answer or invent conditions.
}

Principles:
1. Task feasibility assumption — Only consider the task completable when sufficient information is collected; do not abandon the task prematurely.
2. Strict format — Output only the <reasoning></reasoning> block and the JSON object; no extra text.
3. Goal separation — info_to_collect represents information goals, not procedural steps.
4. Constraint completeness — Ensure that info_to_collect covers every explicit constraint not yet confirmed for task completion, and may include derived constraints to enhance decision quality.
5. Synonyms and variants — Common synonyms/aliases/related terms should be merged within a single item using parentheses or slashes; consolidate overlapping goals to keep the list concise.
6. Preserve key items — When generating a new info_to_collect, retain previously important but unverified items.
7. Remove irrelevant items — Move verified or newly proven irrelevant/useless items into info_not_to_collect.
8. Early termination — If the collected knowledge is sufficient to answer the question or complete the task, mark complete = true as soon as possible. Do not over-focus on unspecified details.
9. Termination conditions — For specific questions: terminate once all conditions are met (avoid excessive exploration of overly detailed sub-information). For open-ended questions: terminate once sufficient basic information to answer is collected.
10. Answer focus — There is no need to provide an overly comprehensive answer beyond the stated conditions; avoid excessive exploration or focusing on user-unstated details.

Figure 7: Prompts for Task Planner in WEBTRAVELER.

Prompt for Perceiver

You are a world-class information extraction agent. Your task is to analyse the given observation content and extract information relevant to the current user query.

Goal:
- You need to produce a brief summary of the current page, i.e., brief_summary.
- You need to determine whether the observation contains new information that is useful for the query, i.e., new_relevant_facts.

Input:
- Query: "<user's question>"
- Info_to_collect: "<current information-gathering goals>"
- Info_not_to_collect: "<current information points that do not need to be collected>"
- Current Collected_information: "<information already collected>"
- Observation: "<current webpage observation content>"

Output format:
<reasoning>
Briefly reason from the following aspects:
- How to extract information from the Observation that is relevant to the Query and Info_to_collect.
- How to avoid collecting information listed in Info_not_to_collect.
- How to avoid re-collecting information already present in Collected_information.
- How to identify all potentially useful clickable buttons while avoiding retrieving too many irrelevant ones.
</reasoning>
{
"brief_summary": "A concise summary of the current page",
"clickable_button_candidates": "A list of up to 5 potentially useful clickable button candidates",
"new_relevant_facts": "Complete sentences describing task-relevant facts, either as a string or a string array"
}

Principles:
1. Strict format — Output only the <reasoning></reasoning> block and the JSON object; no extra text.
2. Memory-oriented — brief_summary should summarize the important and potential information on the page to facilitate recalling this page's usefulness for completing the user's question.
3. Button candidate selection — clickable_button_candidates should include all clickable buttons on the page that have potential to help complete the task. If fewer than five, you may return fewer or even an empty list.
4. Observation priority — new_relevant_facts must have visible support within the current observation; if none, return [].
5. Relevance first — Include only facts that directly (or partially) satisfy info_to_collect.
6. Fact-only — Do not speculate, plan, or hypothesize; avoid vague possibilities.
7. Direct or implied — Facts must be explicitly presented in or logically implied by the observation.
8. Atomic and precise — Each fact must be self-contained and include clear labels/units/timestamps when visible.
9. No repetition — Do not repeat the task, Info_to_collect, or general context. Do not repeat facts already in Collected_information (semantic equivalents count as duplicates), and avoid redundancy within this output.
10. Source awareness — Facts should be tied to visible clues in the observation; include short citations or positions only in <reasoning>, not in the JSON.
11. Source expression — Prefer to use the original wording from the source, with minimal modification; summarisation or synthesis is allowed only for clarity or completeness.
12. Conflict handling — If the observation contradicts prior knowledge, prioritise the observation and output only facts supported by it (do not restate old facts).

Figure 8: Prompts for Perceiver in WEBTRAVELER.

Prompt for Explorer

You are a world-class browser explorer. Your main responsibility is to design the optimal next-step browser exploration strategy based on the current page state, user task, and historical strategies — to efficiently advance the task goal.

Goal:
- First, evaluate the previous browser exploration strategy and reflect on its usefulness and efficiency toward the goal.
- Combine the user query, browser exploration trajectory memory, previous observation and strategy, current page observation, and other relevant information to output a clear and executable next-step browser exploration strategy.
- The exploration strategy must be built using the available toolset; only one tool may be used per step.

Available tools:
tool_descs

Input:
- Query: "<user's question>"
- Info_to_collect: "<current information-gathering goals>"
- Info_not_to_collect: "<current information points that do not need to be collected>"
- Collected_information: "<key information already collected (in bullet form)>"
- Memory of the browser exploration trajectory: "<hierarchical memory of browser exploration trajectory: records exploration policy, observations, and reflections>"
- Observations of the last step: "<observation from the previous step>"
- Last step policy: "<previous strategy>"
- Current Observations: "<current observation>"
- Visited buttons: "<buttons already visited>"
- Previous potential buttons: "<buttons with potential to yield valuable information>"

Output format:

<reasoning>
Briefly reason from the following aspects:
- What is the current highest-priority unsolved goal?
- Based on the Memory of the browser exploration trajectory, are there unexplored new goals or completed ones? How can the exploration path be optimized?
- Based on Visited buttons, how can repeated exploration of the same button be avoided?
- Based on Previous potential buttons, how can comprehensive and efficient exploration of potential content be ensured?
</reasoning>

<reflection>
{
"reflection": "In one or two sentences, evaluate the contribution and efficiency of the previous strategy toward the goal, reflect on the reasons, and propose one concise corrective suggestion for the next step; if necessary, explicitly mention actions that should not be taken."
}
</reflection>

<tool_call>
{
"name": "tool_name",
"parameters": { ... }
}
</tool_call>

Principles:
1. Strict format — Output only the <reasoning></reasoning>, <reflection></reflection>, and <tool_call></tool_call> blocks with JSON objects; no extra text.
2. Evidence-based — Use concrete clues and memory to generate reflections and next-step strategies.
3. Avoid info_not_to_collect — When designing the next action, avoid collecting information that has been verified or deemed irrelevant.
4. Improve efficiency through strategy

history — When formulating the next strategy, refer to the Memory of the browser exploration trajectory; if a similar or identical strategy has already been executed on this page, do not repeat it. Instead, change the target content or interaction type.
5. Improve efficiency through visited buttons — Refer to Visited buttons to avoid exploring the same button again; prioritize unvisited or new buttons with potential, or deepen exploration of the currently explored button.
6. Improve efficiency through potential buttons — Use Previous potential buttons to enhance the completeness and efficiency of browser exploration.

Figure 9: Prompts for Explorer in WEBTRAVELER.

D Potential Risks

The techniques proposed in this paper aim to enhance the robustness and efficiency of WebAgent exploration, and may therefore amplify the associated capabilities in both benign and malicious settings. Such risks are a commonly accompanying dual-use risk of web agent technologies that operate on live websites via automated in-site navigation and evidence aggregation. Once endowed with more stable long-horizon interaction and more efficient information extraction, the barrier to using them for inappropriate automated collection or circumventing access constraints may be correspondingly lowered.