Navigating the Infinite Dynamic Web Space: Effective In-Context Exploration via Cognitive Multi-Agent Collaboration

Guozhao Mo[1,2], Yanjiang Liu[1,2], Yafei Shi[3], Jiawei Chen[1,2], Yang Li[3], Yaojie Lu[2]*, Hongyu Lin[2], Ben He[1,2], Le Sun[2], Bo Zheng[3]*, Xianpei Han[2]

[1]University of Chinese Academy of Sciences, Beijing, China
[2]Chinese Information Processing Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing, China
[3]MYbank, Ant Group
{moguozhao2024,luyaojie}@iscas.ac.cn guangyuan@mybank.cn

Abstract

Dynamic web navigation is challenging due to infinite decision space and the constantly changing nature of cyberspace. Existing methods rely on greedy strategies or value estimation, struggle to achieve effective backtracking and are heavily dependent on proprietary models. In this paper, we propose HINTNAVIGATOR, a cognitive multi-agent collaboration framework that enhances cyberspace exploration capability through In-Context Exploration (ICE). Inspired by the human cognitive planning process, we categorize the interaction history into *Declarative History* (environment observations) and *Procedural History* (action trajectories) to enhance historical reflection capability. These dual-history streams are dynamically integrated through specialized cognitive agents, enabling effective self-directed backtracking guided by working memory consolidation. Experiments show that HintNavigator achieves state-of-the-art performance among open-source LLM agents, surpassing proprietary model Claude-3.5 Sonnet on the WebArena benchmark.

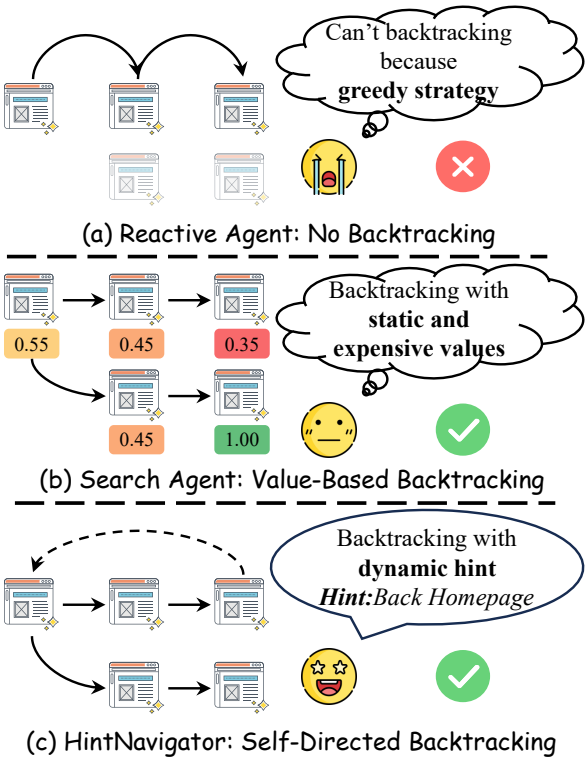

Figure 1: Comparison of Exploration Strategies.

1 Introduction

Large Language Model (LLM) driven agents have made significant progress in planning and reasoning, showing great potential as human assistants (Liu et al., 2024; Wang et al., 2024b; Valmeekam et al., 2023). As cyberspace expands and evolves, users must navigate through overwhelming and continuously changing information to find valuable content efficiently. Dynamic web navigation tackles this challenge by enabling web agents to autonomously explore dynamic web pages, accurately locate target information, and effectively complete user tasks in real time (He et al., 2024).

The key challenges for dynamic web navigation are that web agents must navigate in an ***infinite decision space*** while adapting to a ***constantly changing environment***. The unbounded nature of the web presents agents with an overwhelming number of navigation choices (Baeza-Yates and Castillo, 2007), significantly increasing decision complexity and risking inefficient paths or dead ends. Furthermore, the dynamic nature of web compounds this challenge: after interacting with a link, pages often update content, altering their available choices. These dual challenges of infinite decision-making and environmental dynamism underscore the complexity of developing robust web navigation systems.

Existing approaches for dynamic web navigation can be broadly categorized into two groups: *Reactive Agents* and *Search Agents*. *Reactive Agents* (Figure 1.a) typically based on the Re-

* Corresponding authors

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics
Volume 1: Long Papers, pages 8218–8232
March 24-29, 2026

Act framework (Yao et al., 2023), focus on selecting locally optimal actions for the current states. However, their effectiveness is limited due to the lack of exploration and the disregard for backtracking (see Table 1). On the other hand, *Search Agents*, which employ tree search methods (Browne et al., 2012), construct state-space trees to facilitate value-based backtracking (Figure 1.b). Despite their potential, they encounter significant challenges due to the vastness of the web pages. The exponential growth of the search space results in substantial computational overhead. Moreover, the prevalence of irreversible actions in web navigation makes state backtracking frequently impractical (Gu et al., 2024). Additionally, these approaches predominantly rely on proprietary models like **GPT-4o** (Hurst et al., 2024) or **Claude-3.5 Sonnet** (Anthropic, 2024a), while the development of open-source alternatives has significantly lagged behind. This reliance on closed models not only limits the reproducibility and extensibility of research but also creates substantial barriers to enter for the broader research community.

In this paper, we propose **HintNavigator** (Figure 1.c), a cognitive multi-agent collaboration framework that addresses the critical challenges of dynamic web navigation. Specifically, we introduce *In-Context Exploration*, a strategy that leverages **Hint**—reflects upon historical actions and provides guidance for subsequent decisions—to help the agent perform self-directed backtracking. This approach enables the agent to efficiently navigate in an infinite decision space by dynamically refining its exploration strategy. To better adapt to the constantly changing web environment, we incorporate a dual-history multi-agent framework, where multiple agents collaborate to adjust to environmental changes. Our approach is inspired by human cognitive planning processes, in which *distinct brain regions specialize in different types of information processing before integrating them* for decision-making. We categorize historical information into two distinct types: *Declarative History*, which captures factual and contextual information, and *Procedural History*, which records the sequence of previous actions. This dual-history structure allows the agent to integrate procedural knowledge with declarative cues, facilitating policy refinement and enabling the agent to dynamically adjust its decision-making.

Experiments on the WebArena (Zhou et al., 2024) demonstrate that HintNavigator achieves state-of-the-art success rate among open-source LLMs and delivers results comparable to proprietary LLMs, highlighting its effectiveness in addressing the challenges of infinite dynamic web navigation.

Agent	Go Back(%)	Goto(%)	Backtracking(%)
SteP	0.89	0.64	7.96
AgentOccam-J	3.39	1.22	15.76
WebArena	0.30	1.69	11.03
Human	-	-	30–50

Table 1: Comparison of Backtracking behavior in web navigation: Reactive agent *vs.* Human (See detailed description in Appendix B). Go Back, Goto, and overall Backtracking ratios derived from public trajectories and empirical human browseing data (White and Drucker, 2007; COCKBURN and MCKENZIE, 2001).

The main contributions of this paper are summarized as follows:

- We design an exploration strategy, *In-Context Exploration*, which incorporates self-directed backtracking through *Hint*, significantly improving the efficiency and accuracy in infinite decision space.
- We propose a dual-history multi-agent framework for reflecting on historical information, enabling the agent to dynamically adjust its decision-making in a constantly changing environment.
- We demonstrate that HintNavigator, built on open-source LLMs, not only achieves state-of-the-art performance for open-source models but also delivers performance comparable to proprietary LLMs.

2 Methodology

In this paper, we introduce **HintNavigator** (Figure 2), a cognitive multi-agent collaboration framework designed to enhance dynamic web navigation within infinite decision spaces through in-context exploration. Our work is motivated by the observation that existing LLM driven agents often prioritize locally optimal decisions at each step, which would lead to myopic behaviors. Specifically, once an agent enters an incorrect navigation path, it tends to delve deeper indefinitely, unlike humans who naturally employ iterative exploration and backtracking to identify optimal routes (Table 1).

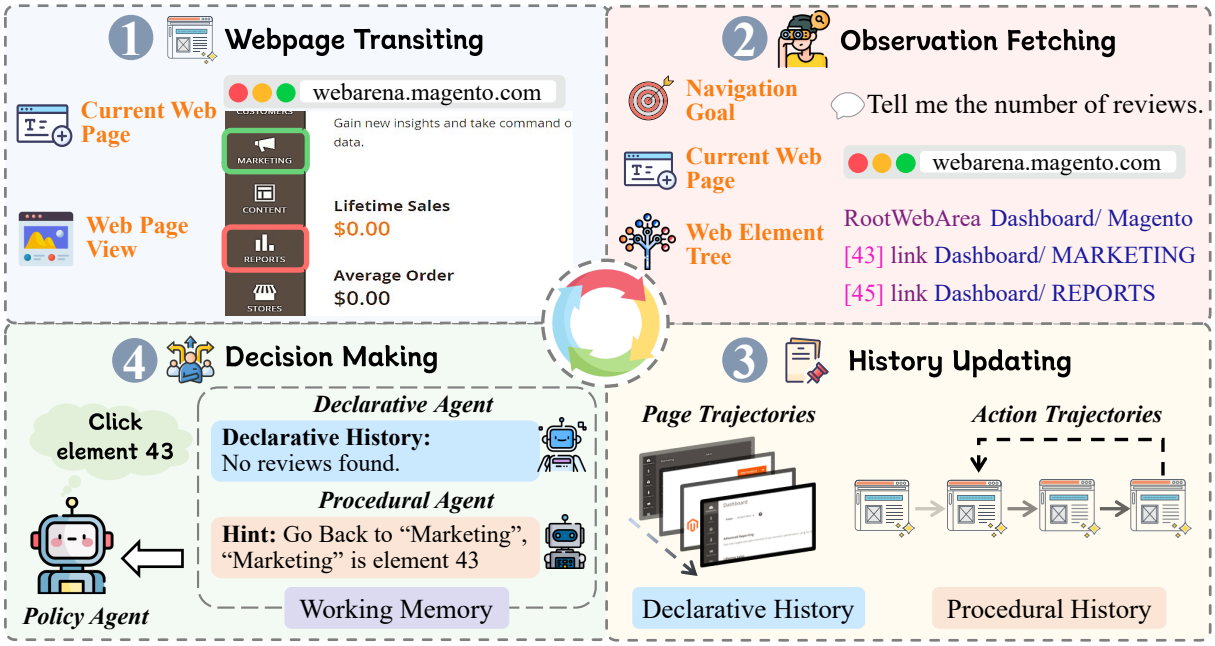

Figure 2: Overview of HINTNAVIGATOR framework. The multi-agent system collaboratively maintains and reflects on history, enabling self-directed backtracking through in-context exploration with dynamic hint generation.

We argue that this limitation stems from the uniform treatment of all historical information in prior approaches, which lack mechanisms for reflective reasoning. To address this, we draw inspiration from the human cognitive architecture ACT-R (Anderson, 1983, 1993). ACT-R is a cognitive architecture theory that explains human cognition and memory mechanisms, providing a framework for understanding how brain processes information and executes corresponding actions. At its core, ACT-R establishes a critical dichotomy in knowledge representation: declarative knowledge (scattered information within observations) versus procedural knowledge (past actions and the reasoning), which collectively constitute working memory to guide decision-making processes.

Building upon this theoretical foundation, we will introduce in-context exploration (in Sec. 2.2) and cognitive multi-agent collaboration framework (in Sec. 2.3).

2.1 Problem Formulation

We formulate dynamic web navigation as a Partially Observable Markov Decision Process (POMDP, Silver and Veness (2010)), where the agent operates under partial observability limited to browser-rendered information (*e.g.*, HTML).

The web environment is characterized by: (1) hidden state space $\mathcal{S}$; (2) observation space $\mathcal{O}$ containing the navigation goal, the accessibility tree and the URL of current web page ; (3) language-action space $\mathcal{A}$, including actions (*e.g.*, click, goto, etc.) and descriptions; (4) deterministic state transition function $\mathcal{T} : \mathcal{S}_t \times \mathcal{A}_t \to \mathcal{S}_{t+1}$; (5) terminal reward function $\mathcal{R} : \mathcal{S} \to \mathbb{R}$ quantifying task completion success.

In this work, we aim to enhance the performance of a fixed LLM policy π_{LLM}. We propose to achieve this through an optimal transformation function $f(\cdot)$ that processes the historical context $\mathcal{H}$. Specifically, we seek to maximize the expected reward of termination state by the policy agent $\pi_{\text{LLM}}(\mathcal{A}|\mathcal{O}, f(\mathcal{H}))$.

2.2 In-Context Exploration in Web Space

Dynamic web navigation is an iterative process in which a policy agent, starting with a specific goal and an initial webpage, chooses an action from a set of actions. This decision is informed by the goal, the webpage's URL, and the accessibility tree, which serves as the observation space. The selected action triggers a change in the webpage, thereby refreshing the observation and repeating the cycle. This process continues until the agent determines an endpoint and stops navigating.

For instance in Figure 2, consider the goal "Tell me the number of reviews". On the initial homepage, the agent selects to click on "REPORTS". This leads to an incorrect page, and for previous reactive agents, finding reviews would be impos-

sible since they only move forward. However, with HintNavigator, guided by a *Hint* (introduced in Sec. 2.3.1), the policy agent generates a backtracking action (goto and go back), enabling self-directed backtracking. This allows the agent to return to the homepage, now aware that clicking "reporting" was a misstep. The agent is guided by hint "Go back to Marketing" then selects to click on "MARKETING", ultimately successfully locating the user reviews. We term this exploration strategy In-Context Exploration.

2.3 Cognitive Multi-Agent Collaboration

Existing LLM-driven agents often treat historical information uniformly, relying solely on the inherent context-handling capabilities of LLMs without distinguishing between different types of history.

This approach, where all historical information is directly fed into policy agent without additional reflection, is the primary reason why existing agents struggle to self-directed backtrack and recover from error paths. To address these limitations, we draw inspiration from the ACT-R cognitive architecture and categorize historical information ($\mathcal{H}$) into two distinct types: *declarative history* ($\mathcal{D}$) and *procedural history* ($\mathcal{P}$).

$$\mathcal{H} = \mathcal{D} + \mathcal{P} \tag{1}$$

Declarative history refers to the scattered information within the observation space. It is information directly encode from the environment and does not require much synthesization (Yengin and Ince, 2014). It emphasizes the what—the explicit facts relevant to the task. For instance, it may involve the memorization and retrieval of HTML snippets or other structured data. By capturing the environmental context, declarative history plays a pivotal role in informing the decision-making processes of the agent, enabling it to leverage explicit, structured knowledge for task resolution.

Procedural history refers to the agents past actions and the reasoning underlying its decisions. It is information encoded from synthesizing and observing transformations of the environment (Yengin and Ince, 2014). Unlike declarative history, which focuses on explicit facts, procedural history emphasizes the why—the rationale behind specific actions or strategies. However, since these actions and reasoning steps may not always be accurate or optimal, they should not be directly incorporated into decision-making without careful evaluation. Instead, procedural history serves as a reflective tool, enabling the agent to learn from past experiences and refine its future behavior.

2.3.1 Hint Generation

We propose to leverage *hint* to guide the policy agent in performing in-context exploration. These hints are designed to provide actionable guidance, and thus, we generate them using *procedural history* composed of action sequences. Specifically, we employ a *procedural agent* (ψ) to generate these hints, ensuring that they are both contextually relevant and actionable for the policy agent.

$$Hint = \psi(\mathcal{P}) \tag{2}$$

Hints guide the policy agent in its next steps. This next step guidance distinguishes our approach from orchestration-based methods, which typically rely on pre-defined global plans rather than dynamic, fine-grained hints.

2.3.2 Working Memory Representation

Relying solely on *hint* leads to the loss of observational information from the *declarative history*, particularly regarding scattered information that is distributed across the observation space. To preserve this crucial information, we introduce a *declarative agent* (ϕ) that maintains these observations and then integrate with hint to form a comprehensive working memory ($\mathcal{W}$) for *policy agent* $\pi_{\text{LLM}}(\mathcal{A}|\mathcal{O}, \mathcal{W})$.

$$\mathcal{W} = \phi(\mathcal{D}) + Hint \tag{3}$$

Theoretically, $\mathcal{D}$ should encompass the entire observation space that has been visited. However, due to the constraints imposed by the limited context window of LLMs, we adopt an iterative approach to update $\mathcal{D}$. Consequently, the updates to $\mathcal{D}$ and $\mathcal{P}$ are formulated as follows:

$$\begin{aligned} \mathcal{D}_t &= \phi(\mathcal{O}_t, \mathcal{D}_{t-1}) \\ \mathcal{P}_t &= \mathcal{P}_{t-1} + (action_t, think_t) \end{aligned} \tag{4}$$

Algorithm 1 presents the pseudocode for HintNavigator. The *Execute* is a pre-defined program that executes the action in the environment. The *TerminationCheck* function determines whether the agent should terminate by evaluating two conditions: (1) if the policy agent generates a send_msg_to_user action, or (2) if the maximum number of t is reached. All prompt templates used in HintNavigator are provided in Appendix J.

Algorithm 1: HintNavigator

Input: Initial observation $\mathcal{O}_0$
Output: Reward $\mathcal{R}$

$t \leftarrow 0$;
$\mathcal{D}, \mathcal{P} \leftarrow \{\}$;
while *True* **do**
  $\mathcal{D} \leftarrow \texttt{DeclarativeAgent}(\mathcal{O}_t, \mathcal{D})$;
  $Hint \leftarrow \texttt{ProceduralAgent}(\mathcal{O}_t, \mathcal{P})$;
  $\mathcal{W} \leftarrow \mathcal{D} + Hint$;
  $\mathcal{A}_t, T_t \leftarrow \texttt{PolicyAgent}(\mathcal{O}_t, W)$;
  $\mathcal{O}_{t+1} \leftarrow \texttt{Execute}(\mathcal{A}_t)$;
  $\mathcal{P} \leftarrow \mathcal{P} + < \mathcal{A}_t, T_t >$;
  $t \leftarrow t + 1$;
  if $\texttt{TerminationCheck}() = True$ **then**
    break;
  end
end
Return $\mathcal{R}(\mathcal{S}_t)$;

3 Experiments and Analysis

3.1 Setup

Web Environment. We evaluate our approach on two web interaction benchmarks: WebArena (Zhou et al., 2024) and Online-Mind2Web (Xue et al., 2025). **WebArena** is a locally deployed benchmark for dynamic web interaction, designed to avoid real-world issues such as anti-crawling mechanisms. It contains 812 tasks across four fully functional websites (Shopping, CMS, Reddit, and GitLab), several utility sites (*e.g.*, Maps, Calculator, and Encyclopedia), and one multi-site collaboration task. **Online-Mind2Web** includes 300 tasks on real-world websites, providing a more realistic evaluation than simulated platforms.

Evaluation Metric. Evaluations are conducted once at task completion, with task success rate (**SR**) serving as the primary performance metric. To determine whether a task is successfully completed, WebArena provides evaluation functions that assess task success through a set of predefined procedures.

Agent Backbone. For the policy agent, we implement a ReAct agent base on Qwen-2.5 72B Instruct (Qwen et al., 2025) and Llama-3.1 70B Instruct (Dubey et al., 2024).

For the action space, prior work by Yang et al. (2025) has demonstrated that the selection of the action space plays a critical role in determining the performance of such agents. Building on their findings, we adopt a simplified action space that maintains the agents full behavioral capabilities while minimizing the gap with pre-trained tasks (see Appendix A for details). This design choice ensures that the agent can operate effectively without requiring extensive task-specific fine-tuning.

For the observation space, it is essential to incorporate as much information as possible from the environment to maximize the agents performance. To this end, we utilize an accessibility tree augmented with webpage URL information. The accessibility tree, a simplified representation of the DOM tree, preserves the structural and semantic clarity of web elements while reducing unnecessary complexity. This approach ensures that the agent has access to rich, structured information about the environment, enabling more informed decision-making.

Baselines. We compare HintNavigator against both proprietary and open-source LLM-based approaches. Specifically, we compare against the following baselines in Webarena: (1) TreeSearch (Koh et al., 2024), which utilizes tree search with backtracking for task execution; (2) AgentLab (Chezelles et al., 2024), which reports results for multiple models under a unified reactive agent; (3) SteP (Sodhi et al., 2024), a method that manually stratifies task-solving strategies; (4) AWM (Wang et al., 2024c), a method that finds trajectory patterns to enhance memory; (5) WebPilot (Zhang et al., 2025), a multi-agent system with strategic exploration; and (6) AgentOccam-J (Yang et al., 2025), a proprietary LLM-based method representing the current state-of-the-art.

On the Online-Mind2Web, we compare our approach with two representative systems: Claude Computer Use (Anthropic, 2024b), a closed-source web automation agent, and Browser Use (Team, 2024), an open-source browser-control framework. These baselines provide a comprehensive comparison across diverse methodologies, highlighting the strengths and limitations of HintNavigator.

3.2 Main Results

We conducted a comprehensive evaluation of HintNavigator by comparing it against both proprietary and open-source LLMs to demonstrate the effectiveness, with the results presented in Table 2-3.

Agents	Model	SR	Shop.	CMS	Red.	Git.	Map	Mul.
Proprietary LLMs								
Tree Search (Koh et al., 2024)	GPT-4o	19.2	-	-	-	-	-	-
SteP (Sodhi et al., 2024)	GPT-4 Turbo*	33.3	33.2	32.4	52.8	26.7	35.8	12.5
AWM (Wang et al., 2024c)	GPT-4	35.5	-	-	-	-	-	-
WebPilot (Zhang et al., 2025)	GPT-4o	37.2	-	-	-	-	-	-
AgentLab (Chezelles et al., 2024)	Claude-3.5 Sonnet	36.2	-	-	-	-	-	-
AgentOccam-J (Yang et al., 2025)	GPT-4 Turbo	**45.7**	**43.3**	**46.2**	**67.0**	**38.9**	**52.3**	**16.7**
Open-sourced LLMs								
AgentLab (Chezelles et al., 2024)	Llama-3.1 70B	18.4	-	-	-	-	-	-
AgentLab (Chezelles et al., 2024)	Qwen-2.5 72B**	15.5	23.0	13.2	8.5	16.7	14.7	8.3
AgentOccam-J (Yang et al., 2025)	Qwen-2.5 72B**	28.6	33.2	23.1	**60.4**	22.8	15.6	**12.5**
HintNavigator (***ours***)	Llama-3.1 70B	27.2	26.2	24.7	41.5	30.6	22.9	6.3
HintNavigator (***ours***)	Qwen-2.5 72B	**36.5**	**35.3**	**41.2**	51.9	**39.4**	**22.9**	8.3

Table 2: Experiment results on WebArena. All open-source LLMs refer to their Instruct versions. * denotes that we obtain their detailed results from AgentOccam. ** denotes that we rerun their code with specific LLM.

Method	Model	SR (%)
HintNavigator (***ours***)	Qwen-2.5 72B	**28.7**
Browser Use (Team, 2024)	GPT-4o	24.3
Computer Use (Anthropic, 2024b)	Claude-3.5 Sonnet	26.0

Table 3: Experiment results on Online-Mind2Web. We use `WebJudge-7B` for evaluation.

HintNavigator achieves state-of-the-art performance among open-source LLM agents. HintNavigator attaining a success rate of 36.5% with the Qwen-2.5 72B Instruct model in WebArena, surpassing the results of the current SOTA method, AgentOccam-J, when using the same model. With Llama-3.1 70B Instruct, HintNavigator achieves a success rate of 27.2%, a 47.8% relative improvement over AgentLab using the same model. Notably, HintNavigator, utilizing the Qwen-2.5 72B Instruct model, surpasses the performance of the Claude-3.5 Sonnet in both benchmarks.

HintNavigator Excels in Single-Site Tasks with Room for Multisite Enhancement. HintNavigator demonstrates strong performance on Reddit (51.9%) and Gitlab (39.4%) tasks, showcasing its effectiveness in diverse web interactions, while its lower success rate on Multisite tasks (8.3%) highlights potential areas for future improvement.

3.3 Ablation Studies

3.3.1 Impact of Working Memory

To investigate the impact of working memory, we conducted ablation studies on HintNavigator under various working memory configurations in WebArena, with the results presented in Table 4.

HintNavigator achieves the optimal working memory configuration. We observe substantial performance gains across both Qwen and Llama. This performance improvement is primarily attributed to single-website tasks, where the model benefits from focused contextual cues. However, the advantage diminishes in multi-website, which we attribute to potential interference from cross-website declarative history that may introduce cognitive conflicts. Overall, HintNavigator demonstrates consistent performance advantages.

Hint plays a crucial role in working memory. Specifically, using $Hint$ alone achieves suboptimal performance, which is notably superior to using only $\mathcal{P}$, only $\mathcal{D}$, or the combination of $\mathcal{D} + \mathcal{P}$. Furthermore, the use of $Hint$ alone yields the best success rate in multi-website tasks, highlighting their effectiveness in cross-domain scenarios. These results suggest that the role of $\mathcal{D}$ may be limited in certain contexts, warranting further investigation in future work to better understand its scope and applicability.

Using history without distinction leads to suboptimal performance. Specifically, when $\mathcal{D} + \mathcal{P}$ are directly incorporated into the working memory, we observe performance degradation compared to using $\mathcal{D}$ alone, as evidenced by experimental results on the Llama. Furthermore, our

Experimental Settings	SR	Shopping	CMS	Reddit	Gitlab	Map	Multisite
Qwen-2.5 72B Instruct							
HintNavigator	**36.5**	**35.3**	**41.2**	**51.9**	**39.4**	22.9	8.3
$\mathcal{W} = Hint$	30.1	33.7	30.2	41.5	30.6	19.3	**12.5**
$\mathcal{W} = Declarative\ History$	27.6	31.6	22.0	41.5	25.6	**27.5**	10.4
$\mathcal{W} = Procedural\ History$	26.4	34.2	19.8	41.5	26.7	16.5	8.3
$\mathcal{W} = Decl. + Proc.\ History$	28.5	30.5	27.5	46.2	26.7	21.1	8.3
Llama-3.1 70B Instruct							
HintNavigator	**27.2**	**26.2**	24.7	**41.5**	**30.6**	**22.9**	6.3
$\mathcal{W} = Hint$	25.9	26.2	**26.4**	36.8	25.0	21.1	**12.5**
$\mathcal{W} = Declarative\ History$	25.1	25.1	24.7	31.1	27.8	22.9	8.3
$\mathcal{W} = Procedural\ History$	22.2	24.6	20.3	37.7	21.1	14.7	6.3
$\mathcal{W} = Decl. + Proc.\ History$	24.4	25.7	23.1	32.1	26.7	21.1	6.3

Table 4: Component-wise analysis of HintNavigator: Evaluating the contribution of history and hint.

Agent	Go Back(%)	Goto(%)	Backtracking(%)	SR
AgentLab*	0.09	3.22	6.90	16.0
AgentOccam-J*	3.46	**4.93**	24.80	28.6
HintNavigator	3.46	3.80	**40.76**	**36.5**
$\mathcal{W} = Hint$	2.91	3.30	36.95	30.1
$\mathcal{W} = \mathcal{D}$	2.94	2.56	12.81	27.6
$\mathcal{W} = \mathcal{P}$	**4.61**	2.27	23.52	26.4
$\mathcal{W} = \mathcal{D} + \mathcal{P}$	3.18	4.61	39.04	28.5
Human	-	-	30–50	78.2

Table 5: Comparative analysis of backtracking in Qwen. $\mathcal{W}$ denotes working memory. * denotes that we rerun their open-sourced code with Qwen.

findings demonstrate that the direct application of $\mathcal{P}$ yields inferior outcomes compared to $hint$, strongly validating the rationale behind adopting $Hint$ as an alternative to $\mathcal{P}$.

3.3.2 Impact of Backtracking

To investigate the impact of backtracking, we conducted a quantitative analysis of backtracking-related actions on the Qwen-2.5 72B Instruct. The detailed results are presented in Table 5, and the complete action frequency statistics are illustrated in Figure 3. This analysis provides insights into the frequency and necessity of self-directed backtracking in our framework, which is crucial for understanding the model's navigation behavior. Specifically, we measured two metrics: (1) the proportion of backtracking actions (including Go Back and Goto) in all actions. (2) the percentage of trajectories that including backtracking action.

HintNavigator enhances the backtracking rate. In evaluations conducted on the same Qwen model, HintNavigator exhibits a 15.96% increase in backtracking rate over AgentOccam-J, the state-of-the-art reactive agent. This improvement underscores the effectiveness of HintNavigator in navigating infinite decision spaces by leveraging in-context exploration strategy for error recovery.

Hint is the key to backtracking. Under only $Hint$, agent achieves 36.95% backtracking ratio, outperforming only $\mathcal{D}$ (12.81%) and $\mathcal{P}$ (23.52%).

3.4 Case Study

To investigate the advantages and limitations of In-Context Exploration, we conducted a qualitative analysis on several instances of HintNavigator.

Effectiveness of In-Context Exploration. In-context exploration can guide the agent to recover from error paths. Given the infinite decision space and the dynamic nature of web environment, an agent is prone to being led astray once it makes suboptimal decisions. In-context exploration enables the agent to recognize when it has deviated from the correct path through reflective analysis of its history. By leveraging hints, the agent can perform self-directed backtracking and realign itself with the correct path. One example is Task #13 (see in Appendix G), where the agent initially diverges into an incorrect path but successfully reverts to the correct path with the aid of hints.

Error Analysis. We conducted a manual analysis of 100 error cases sampled from the trajectories generated by HintNavigator in Qwen-2.5 72B Instruct, with the results illustrated in Figure 4. The

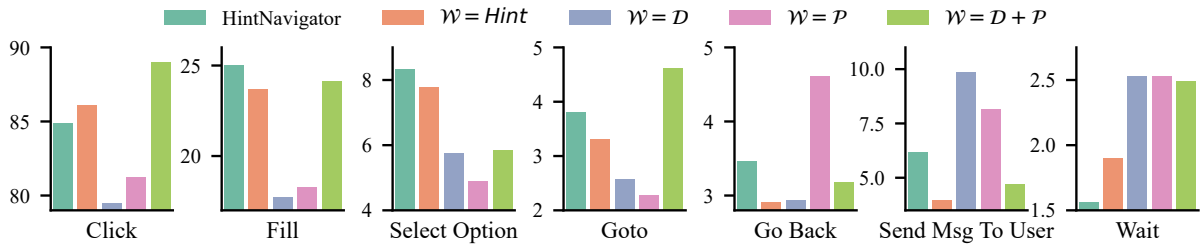

Figure 3: Complete action frequency (%) statistics in Qwen-2.5 72B Instruct.

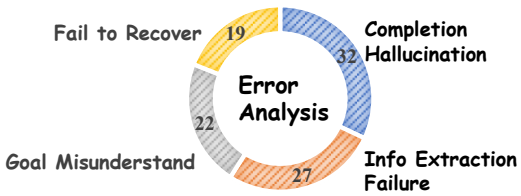

Figure 4: Error type distribution of HintNavigator of 100 incorrect trajectories.

errors were categorized into four types: (1) Completion Hallucination, where the agent incorrectly assumes that the goal has been either completed or not completed. For example, the agent might prematurely conclude that a user's request to "book a flight" is completed after selecting a flight, without proceeding to the payment confirmation page; (2) Information Extraction Failure, where the agent fails to extract all answers or information from the page. For instance, when asked to find the price of a product on an e-commerce page, the agent might only extract the price missing the additional fees such as shipping costs; (3) Goal Misunderstanding, where the agent misinterprets the intended goal. For example, the user's goal is "What is the top-1 best-selling product in 2022", the agent might misinterpret this as searching for best-selling product, regardless of year; (4) Fail to Recover, where the agent is unable to identify the optimal path to achieve the goal. For instance, when navigating a complex website to complete a multi-step task like applying for a visa, the agent might get stuck in a loop of revisiting the same pages.

4 Related Work

Reactive Agents for Dynamic Web Navigation. The development of reactive agents has progressed in tandem with proprietary LLMs. These agents operate reactively, making decisions based on immediate observations without extensive planning, often using frameworks like ReAct (Yao et al., 2023). A significant body of work focuses on designing specialized modules to decompose complex tasks (Sodhi et al., 2024; Pan et al., 2024a; Drouin et al., 2024), leveraging proprietary models such as GPT-4 or GPT-4o. Other approaches improve performance by extracting patterns from historical trajectories (Wang et al., 2024c). However, reliance on proprietary models raises scalability and privacy concerns. In contrast, recent research has demonstrated competitive performance by fine-tuning open-source models on large-scale trajectory data (Lai et al., 2024; Chae et al., 2024; Qi et al., 2025; Su et al., 2025), circumventing the limitations of proprietary systems. In this work, we explore the shared challenges across these paradigms and investigate the potential of open-source models to achieve state-of-the-art performance, aiming to advance accessible and scalable web agents.

Search Agents for Dynamic Web Navigation. Tree search methods have shown promise in helping agents balance exploration and exploitation in environments. These methods typically rely on value functions to estimate state rewards. However, real-world tree search (Koh et al., 2024) often requires state backtracking, which is infeasible due to irreversible operations on websites (Gu et al., 2024). Model-based tree search introduces web-world models to simulate interactions (Gu et al., 2024; Chae et al., 2024), but this approach compounds uncertainty, leading to suboptimal performance. Additionally, value estimation in web is challenging due to the absence of explicit reward signals. To address these limitations, we propose in-context exploration that enables autonomous learning of exploration-exploitation balance. By leveraging procedural history, our method generates dynamic hints to guide agents in escaping suboptimal trajectories, reducing reliance on explicit value functions and mitigating the challenges of ir-

reversible actions. This approach enhances adaptability and decision-making efficiency in complex web environments.

5 Conclusion

In this paper, we introduced HintNavigator, a Cognitive Multi-Agent Collaboration Framework using In-Context Exploration strategy to address the challenges of infinite dynamic web navigation. Inspired by the ACT-R, we categorize history into declarative history and procedural history, employing two specialized agents to generate working memory and guide the policy agent through a self-directed backtracking mechanism using hints. The results on the WebArena demonstrate the effectiveness of HintNavigator. Our findings not only advance the state-of-the-art in open-source LLM-based web navigation but also provide a foundation for future research in cognitive-inspired multi-agent systems for interactive tasks.

Limitations

We acknowledge the limitation of our study that we aim to address in future work.

Limited exploration in training LLMs specifically for dynamic web navigation. Our study primarily focused on utilizing pre-trained, static LLMs without further fine-tuning or specialized training. We hypothesize that incorporating large-scale training using web navigation trajectories could potentially enhance HintNavigator's performance by better capturing the dynamic nature and state-dependent decision processes inherent in real-world web environments.

Acknowledgements

We sincerely thank the reviewers for their insightful comments and valuable suggestions. This work was supported by the Natural Science Foundation of China (No. 62272439, 62306303, 62476265) and Ant Group, MYbank.

References

John R. Anderson. 1983. *The Architecture of Cognition*. Harvard University Press.

John R Anderson. 1993. Problem solving and learning. *American psychologist*, 48(1):35.

Anthropic. 2024a. Introducing claude 3.5 sonnet.

Anthropic. 2024b. Introducing computer use, a new claude 3.5 sonnet. https://www.anthropic.com/news/3-5-models-and-computer-use. 2025-10-05.

Ricardo Baeza-Yates and Carlos Castillo. 2007. Crawling the infinite web. *Journal of Web Engineering*, pages 049–072.

Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43.

Hyungjoo Chae, Namyoung Kim, Kai Tzu iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. 2024. Web agents with world models: Learning and leveraging environment dynamics in web navigation. *Preprint*, arXiv:2410.13232.

Thibault Le Sellier De Chezelles, Maxime Gasse, Alexandre Drouin, Massimo Caccia, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Quentin Cappart, Graham Neubig, Ruslan Salakhutdinov, Nicolas Chapados, and Alexandre Lacoste. 2024. The browsergym ecosystem for web agent research. *Preprint*, arXiv:2412.05467.

ANDY COCKBURN and BRUCE MCKENZIE. 2001. What do web users do? an empirical analysis of web use. *International Journal of Human-Computer Studies*, 54(6):903–922.

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 Information Processing Systems*, volume 36, pages 28091–28114. Curran Associates, Inc.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. 2024. WorkArena: How capable are web agents at solving common knowledge work tasks? In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 11642–11662. PMLR.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yu Gu, Boyuan Zheng, Boyu Gou, Kai Zhang, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. 2024. Is your llm secretly a world model of the internet? model-based planning for web agents. *Preprint*, arXiv:2411.06559.

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 *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6864–6890, Bangkok, Thailand. Association for Computational Linguistics.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. 2024. Tree search for language model agents. *Preprint*, arXiv:2407.01476.

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. 2024. Autowebglm: A large language model-based web navigating agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 52955306, New York, NY, USA. Association for Computing Machinery.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, and Percy Liang. 2018. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations*.

Yanjiang Liu, Tianyun Zhong, Yaojie Lu, Hongyu Lin, Ben He, Shuheng Zhou, Huijia Zhu, Weiqiang Wang, Zhongyi Liu, Xianpei Han, and Le Sun. 2024. XMC-agent : Dynamic navigation over scalable hierarchical index for incremental extreme multi-label classification. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5659–5672, Bangkok, Thailand. Association for Computational Linguistics.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. 2024a. Autonomous evaluation and refinement of digital agents. In *First Conference on Language Modeling*.

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

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, Tianjie Zhang, Wei Xu, Jie Tang, and Yuxiao Dong. 2025. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *Preprint*, arXiv:2411.02337.

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, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. 2017. World of bits: An open-domain platform for web-based agents. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3135–3144. PMLR.

David Silver and Joel Veness. 2010. Monte-carlo planning in large pomdps. In *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc.

Paloma Sodhi, S.R.K Branavan, Yoav Artzi, and Ryan McDonald. 2024. Step: Stacked LLM policies for web actions. In *First Conference on Language Modeling*.

Hongjin Su, Ruoxi Sun, Jinsung Yoon, Pengcheng Yin, Tao Yu, and Sercan Ö. Ark. 2025. Learn-by-interact: A data-centric framework for self-adaptive agents in realistic environments. *Preprint*, arXiv:2501.10893.

Browser Use Team. 2024. Browser use. `https://github.com/browser-use/browser-use`. 2025-10-05.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023. On the planning abilities of large language models - a critical investigation. In *Advances in Neural Information Processing Systems*, volume 36, pages 75993–76005. Curran Associates, Inc.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024a. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. In *Advances in Neural Information Processing Systems*, volume 37, pages 2686–2710. Curran Associates, Inc.

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

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2024c. Agent workflow memory. *Preprint*, arXiv:2409.07429.

Ryen W. White and Steven M. Drucker. 2007. Investigating behavioral variability in web search. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 2130, New York, NY, USA. Association for Computing Machinery.

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. *Preprint*, arXiv:2504.01382.

Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. 2025. Agentoccam: A simple yet strong baseline for LLM-based web agents. In *The Thirteenth International Conference on Learning Representations*.

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 *The Eleventh International Conference on Learning Representations*.

Ilker Yengin and Ibrahim Furkan Ince. 2014. Applying the adaptive control of thought-rational theory into the design of mobile worked examples applications. *International Journal of Robots, Education and Art*, 4(2):21.

Yao Zhang, Zijian Ma, Yunpu Ma, Zhen Han, Yu Wu, and Volker Tresp. 2025. Webpilot: A versatile and autonomous multi-agent system for web task execution with strategic exploration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(22):23378–23386.

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 *The Twelfth International Conference on Learning Representations*.

A Action Space

To ensure the agent's ability to comprehensively execute all possible actions on a web page while minimizing the action space, we have carefully optimized our action space. The detailed action space is presented in Table 6.

Action	Description
wait [time]	Wait time
click [id]	Click an element
fill [id] [content]	Fill an element
goto [url]	Goto URL
go_back	Go back of the page
send_msg_to_user	Send message to user

Table 6: Action space for agent to interact with web.

B Description of Action Behavior Analysis

Due to the cyclic graph nature of web, certain click actions may inadvertently cause backtracking behavior. Simply measuring the ratio of explicit "Go Back" and "Goto" actions as the backtracking rate could lead to significant measurement biases.

To obtain accurate statistics, we conducted a meticulous annotation study with three annotators holding Master's degrees in Computer Science. These annotators independently examined the first 100 task trajectories in WebArena. For cases with disagreement, we held consensus meetings to resolve discrepancies through discussion.

The results in Table 7 reveal the key findings: The proportion of backtracking caused by click actions is sufficiently small that it doesn't significantly impact our analysis of the ICE component's contribution. Moreover, browser navigation actions and explicit jumps prove more efficient than click-induced backtracking. This is particularly evident in deep navigation scenarios, where click-based methods (typically limited to top-level navigation elements) are less effective than direct jumps to specific pages.

C Introduction to ACT-R

HintNavigator draws inspiration from the Adaptive Control of ThoughtRational (ACT-R) architecture, a cognitive framework that seeks to explain higher-level human cognitive processes. ACT-R provides a theoretical model of how humans process information and subsequently take action. At

Model	Backtracking w/o click (%)	Backtracking w/ click (%)
AgentOccam-J	16.0	22.0
SteP	5.0	6.0
HintNavigator	43.0	43.0
Human	-	30-50

Table 7: Backtracking ratio with or without click.

its core, ACT-R posits that cognitive processes are grounded in a unified system, meaning that all human thought, regardless of its nature, arises from the same underlying neural mechanisms. This principle bears a striking resemblance to the behavior of LLM-driven agents, particularly when operating within the constraints of a fixed LLM.

One of the central tenets of ACT-R is the distinction between two types of knowledge: declarative knowledge, which encompasses facts or rules for solving mathematical equations, and procedural knowledge, which involves habitual behaviors such as riding a bicycle or driving a car. This dichotomy offers a compelling explanation for the limitations observed in earlier web agents, which often struggled to perform deep exploration of historical information. As a result, these agents lacked the ability to engage in retrospective reasoninga critical exploratory capability that is fundamental to human cognition.

D Benchmarks for Web Navigation

The evaluation of web agents has evolved significantly, driven by the growing demand for automated web tasks powered by LLMs and VLMs. Early benchmarks like MiniWob (Shi et al., 2017) and MiniWoB++ (Liu et al., 2018) established the foundation by using simplified web environments. Subsequent advancements introduced more realistic frameworks, such as Mind2Web (Deng et al., 2023), which employed static snapshots of real-world websites to better simulate web interactions. More recently, WebArena (Zhou et al., 2024) has emerged as a leading benchmark, offering a dynamic evaluation environment by hosting real-world websites on local servers. Extending this paradigm, Mind2Web-live (Pan et al., 2024b) enables evaluations on live websites, though this introduces challenges such as network variability, anti-crawling mechanisms, and dynamic content changes. These factors, while reflective of real-world conditions, can obscure the assessment of an agent's decision-making capabilities. To focus on core decision-making, we adopt WebArena, which provides a realistic yet controlled environment, mitigating issues related to network reliability and anti-crawling measures.

Method	Model	SR (%)
HintNavigator (***ours***)	Qwen 2.5 72B	<u>51.5</u>
HintNavigator (***ours***)	Llama 3.1 70B	39.4
AgentLab (Chezelles et al., 2024)	Llama 3.1 70B	27.9
AgentLab (Chezelles et al., 2024)	Claude 3.5 Sonnet	**56.4**
WorkArena (Drouin et al., 2024)	GPT-4o	42.7

Table 8: Performance comparison on WorkArena.

E Experiments on WorkArena

To evaluate HintNavigator's capability in realistic, complex scenarios, we conducted experiments using the WorkArena (Drouin et al., 2024), which is built on the widely-used ServiceNow platform.

Our results in Table 8 demonstrate that HintNavigator achieves competitive performance across proprietary models. Notably, HintNavigator with Qwen 2.5 72B outperforms GPT-4o in WorkArena.

F Compare to Reflection Agent

F.1 Web Agents

Existing web agents typically employ *trajectory-level* reflection mechanisms. For instance, AutoEval (Pan et al., 2024a) conducts post-episode evaluations to guide subsequent retries. In contrast, HintNavigator operates at *step-level* granularity, maintaining and utilizing contextual information across multiple decision points. Prior attempts at step-level reflection in web environments often failed due to insufficient differentiation of historical information treating webpage history and action history uniformly resulted in noisy signals that hindered effective policy updates.

F.2 GUI Agents

GUI agents like Mobile-Agent-v2 (Wang et al., 2024a) focus on *post-action correction*, modifying the current action after observing its outcomes. HintNavigator instead provides *forward-looking guidance* through hints for subsequent actions. The post-correction paradigm risks agent deadlock when: (1) the correction module fails to diagnose errors accurately, or (2) no valid correction can be derived for the current state. HintNavigator avoids these pitfalls through *self-directed* policy adjustment, where the agent autonomously incorporates hints into its decision-making process without external intervention.

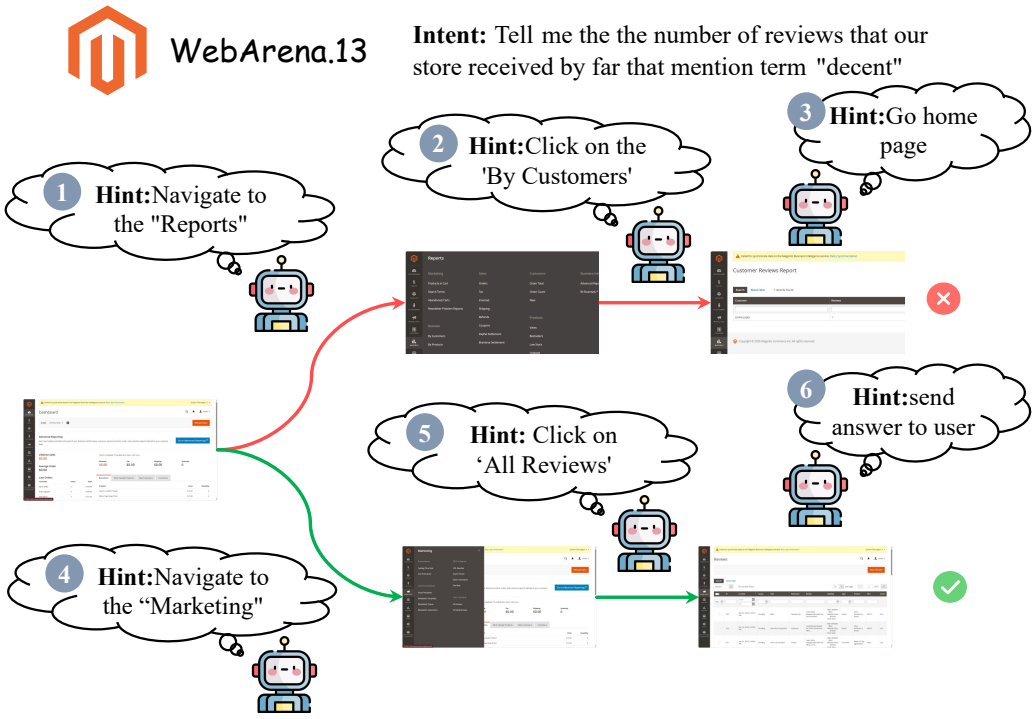

Figure 5: An illustrative example demonstrating HintNavigator’s capability in guiding agents to escape from sub-optimal trajectories through in-context exploration with strategic hint utilization.

G Role of Backtracking

We conduct a detailed analysis of existing methods’ behavior on WebArena Task #13 to demonstrate how the absence of backtracking leads to failure.

The objective of WebArena #13 is to count the number of comments containing ”decent” in CMS. In AgentOccam-J’s trajectory, the agent navigates to the ”Reports” page—a seemingly intuitive but ultimately incorrect path that only provides partial comment information. After a series of subsequent actions, the model concludes that no answer exists (the red path in Figure 5). Without backtracking capability, AgentOccam-J becomes trapped in this wrong navigation branch and returns an incorrect answer. Similar behavioral patterns are observed in AutoEval (using reflection agents) and SteP (with manually designed strategies).

By contrast, HintNavigator demonstrates the effectiveness of hint-guided backtracking. When detecting potential incompleteness in its findings, the system utilizes hints to redirect the agent back to the homepage. This enables correct navigation to the “Marketing” section, where the complete answer is ultimately located (see the green path in Figure 5).

H Error Analysis of Multi-Site Tasks

We find that the success rate of multi-site tasks is lower than that of other task types. To gain deeper insights, we perform a dedicated error analysis. The most frequent error is Completion Hallucination (53.2%), followed by Goal Misunderstanding (19.5%) and Failure to Recover (27.3%). This analysis sheds light on key failure modes and may inform future efforts to improve the performance of multi-site tasks.

I Experimental Rigor

To ensure experimental rigor, we perform comprehensive statistical significance tests to validate our findings.

Given the complex distribution of evaluation metrics in web agent tasks, we apply bootstrap significance testing for a robust analysis of performance differences. Following prior work (Koehn, 2004), we conduct 1,000 bootstrap iterations to assess the significance of metric score differences. The results, shown in Table 9, confirm that HintNavigator consistently outperforms the baseline with statistical significance, reinforcing the credibility of our conclusions.

Agent	Model	SR (%)
HintNavigator (***ours***)	Qwen-2.5 72B	36.5***
HintNavigator (***ours***)	Llama-3.1 70B	27.2***
Agentlab (***baseline***)	Qwen-2.5 72B	15.5

Table 9: Significance tests for main experiments over baseline. ***: $p < 0.005$

J Prompt Templates

Prompt for Abstract Examples

Abstract Examples
Here is an abstract version of the answer with description of the content of each tag. Make sure you follow this structure, but replace the content with your answer:
<think >
Think step by step. The date format should use MM/DD/YYYY, write it down. Describe the effect that your previous action had on the current content of the page.
</think >

Prompt for Procedural Agent

Instructions
You are an assistant agent supporting WebAgent in solving a user-assigned web task. Your role is to evaluate whether WebAgent should explore a new path, as the current path may not be the most efficient. Encourage WebAgent to explore shortcuts and alternative strategies for solving the task more effectively. You should guide the WebAgent go back or goto a new path if it is stuck or inefficient. Provide hints to help WebAgent explore new paths. The hints must can be done in the observation.
Note: Your reply will be interpreted and executed by a program, so make sure to adhere strictly to the formatting requirements. Respond thoughtfully. Only give hints when you think the current action is wrong.

```
Goal
Accessibility Tree
Procedural History
Abstract Examples
```

Prompt for Declarative Agent

Instructions
You are an assistant agent supporting WebAgent in solving a user-assigned web task. Your task is to assist the WebAgent in managing the list of candidate answers by getting potential answers from the observation of current step and integrating them with previous candidate answers. Be careful candidate answer is not the element of the page, but the answer to the question in the goal. Please answer directly to the web task without considering too many possibilities. If the task is ambiguous, understand it the way the user is most likely to understand it. List detailed information to avoid getting the same answer multiple times.
Note: Your reply will be interpreted and executed by a program, so make sure to adhere strictly to the formatting requirements. Respond thoughtfully. Don't give the instruction to WebAgent, just provide information.

```
Goal
Accessibility Tree
Declarative History
Abstract Examples
```

Prompt for Policy Agent

Instructions
You are a UI Assistant. Your goal is to assist the user in performing tasks using a web browser. You can communicate with the user via a chat, where the user provides instructions, and you respond with a single message. After responding, no further interaction from you will occur unless explicitly prompted again.
You have access to a web browser that is visible to both you and the user, but only you can interact with it through specific commands.
Carefully review the users instructions, the current state of the page, and all other relevant information to determine the best possible next action. Ensure that your response is concise, precise, and aligned with the user's request.
Your reply will be interpreted and executed by a program, so make sure to adhere strictly to the formatting requirements. Respond thoughtfully and make your one message count.
Chat messages:
`Chat Messages`

`Accessibility Tree`
`Working Memory`
`Action Space`
`Abstract Examples`